Knowl Inf Syst (2009) 21:327–363
DOI 10.1007/s10115-009-0233-z

REGULAR PAPER

Practical protocol for Yao's millionaires problem enables secure multi-party computation of metrics and efficient privacy-preserving k-NN for large data sets

Artak Amirbekyan · Vladimir Estivill-Castro

Received: 6 September 2007 / Revised: 18 March 2009 / Accepted: 20 June 2009 /
Published online: 17 July 2009

Abstract Finding the nearest k objects to a query object is a fundamental operation for many data mining algorithms. With the recent interest in privacy, 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 have been costly and can only be realistically applied to small data sets. In this paper, we provide efficient solutions for k-NN 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 practical approach to the Yao's millionaires problem with more than two parties. This is based on a pragmatic and implementable solution to Yao's millionaires problem with shares. We also provide privacy-preserving algorithms for combinations of local metrics into a global metric that handles the large dimensionality and diversity of attributes common in vertically partitioned data. To manage very large data sets, we provide a privacy-preserving *SASH* (a very successful data structure for associative queries in high dimensions). Besides providing a theoretical analysis, we illustrate the efficiency of our approach with an empirical evaluation.

Keywords Privacy-preserving data mining · Secure multi-party computation · Nearest-neighbour classification · Yao's millionaires problem

1 Introduction

The diffusion of global threats, like terrorism, requires collaboration and partnership between many governments and/or corporations. Data mining has been identified as one of the most

A. Amirbekyan
Earth Systems Science Computational Centre, The University of Queensland,
Brisbane, QLD 4072, Australia

V. Estivill-Castro (✉)
School of ICT, Griffith University, Brisbane, QLD 4111, Australia
e-mail: v.estivill-castro@griffith.edu.au

useful tools for the fight on terror and crime [41]. However, the information needed resides with many different data holders, and such collaboration may be required among parties that mutually do not trust each other, or between parties that have conflicts of interest. But all parties are aware of the benefits brought by such collaboration. For this kind of collaboration, data privacy becomes extremely important. In the privacy-preserving model, all parties of the collaboration promise to provide their private data to the collaboration, but all want to minimize what the others or any third party may learn about their private data. Collection of data by several agencies results in large databases, those demand efficient methods for data retrieval. For most data mining algorithms, the data is encoded as vectors in high dimensional space.[1] For these algorithms, a measurement of similarity (or dissimilarity) is necessary and often fundamental for their operation. Similarity queries on multi-dimensional data are usually implemented by finding the closest feature vector(s) to the feature vector of the query data. More importantly, in large databases, the high dimensional space can be subject to the curse of dimensionality, and in such settings, 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. This type of query is known as a nearest neighbor (NN) query [46], it has been extensively studied in the past [5,10,15] and constitutes the basis of one of the top 10 algorithms in data mining [61]. A closely related query is the ϵ-range query where all feature vectors that are within the ϵ-neighborhood of the query point q are retrieved.

Similarity search is widely used as a common form of query in modern database applications such as multimedia information systems [50], geographical information systems (GIS) [14], time-series databases [24], medical imaging [38], and bioinformatics [34]. The similarity between two objects is defined with a distance function, e.g., Euclidean distance, between the corresponding feature vectors. For example, in image databases, the query can ask for the most similar images to a given image [4]. 3D shape histograms are used in molecular biology to find similar 3D proteins [3]. Consider a database consisting of DNA sequence of people with cancer. Users can search this database to see if their DNA sequence is similar to the ones in the database. Here privacy-preserving similarity search is very important. While range queries enable distance-based clustering (as in *DBSCAN* with *R*-Trees) and outlier detection [54], k-NN queries also enable Local Outlier Detection [12], Shared Nearest Neighbour Clustering [47] and k-NN Classification [2,35,47].

This wide variety of data mining task, for which k-NN or range queries are a fundamental operation, has recently prompted approaches to privacy-preserving k-NN [2,35,47,54,55]. However, these approaches do have some serious shortcomings. For example, the suite of privacy-preserving algorithms [55] to create a privacy-preserving version of Fagin's *A0* algorithm [23] proved costly even though 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 efficiency. Other common limitations have been the need to compute all pairs of distances [54], to have the query-point public [35], or to deal with horizontally partitioned data [2,47].

In all settings, the additional cost of preserving privacy is non-trivial [36,39,51,59,62]. In this paper we use n for the number of data vectors when this is a small set. In such a case, the additional cost of privacy should be clearly an absolute priority. In a sense, these are relatively few values and releasing information is a large relative loss. We will use N when we are referring to a large dataset, as it is usually the case in data mining applications,

[1] Attribute-vectors are the common input for learning algorithms like decision trees, artificial neural network or for clustering algorithms like k-Means or *DBSCAN*.

and here we accept that the cost of information leak (relative to the dataset size) and the necessity for efficiency justifies a small leakage of information as long as that leakage can be identified as inconsequential and innocuous. Previous work has not produced totally secure algorithms [47,55].

When the dataset is large, data structures that efficiently support k-NN search are essential to many applications. This family includes classic search structures like k-d-trees [8] and R-Trees [29], and newer structures and techniques such as SR-trees [37], X-trees [9] and iDistance [64]. The central role of these indices in data mining algorithms is illustrated by the role that R-Trees play in the efficiency of the popular data mining clustering algorithm *DBSCAN*[1,20].

This paper shows how to compute distances in a privacy-preserving context. This allows privacy-preserving k-NN queries. Finally, we apply protocols for computing distances to a very successful data structure for associative queries in high dimensions, the *SASH*. We emphasize the *SASH* [31] as this data structure makes minimal assumptions about the nature of the metric for associative queries. The *SASH* is neither a spatial index nor a metric index: it makes no assumptions on the nature of the database elements other than the existence of a pairwise distance measure, nor does it require the measure to satisfy the triangle inequality. Houle [31] has shown that for approximate k-NN (k-ANN) queries on very 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. Houle's research also demonstrates that the *SASH* offers better performance, and significantly better control over the time-accuracy trade-off, than previous approximation methods based on metric indices. The *SASH* has already been successfully applied to clustering and navigation of very large, very high dimensional text data sets [30], and spatial data mining of web documents [42]. Two papers [31,32] present the details of the *SASH* structure and its query methods. The *SASH* becomes of particular interest for its potential for data mining because the involvement of many parties results in data of high dimensions that is vertically partitioned. However, the current form of the *SASH* would be inappropriate for privacy preservation. Therefore, this paper develops a *SASH* and its algorithms so that parties are confident that privacy of their data is pragmatically protected. Naturally, the result of k-NN queries by one party on the union of the data reveals information on the other party's private data since significant information can be inferred from the result of the query alone. Privacy is threatened further if queries are repeated or malicious queries are issued. This paper aims at offering a pragmatic (practical and efficient) solution while specifying the departure from some theoretical ideal solutions.

2 Private collaborations

We study collaboration between several parties that wish to compute a function of their collective databases. In fact, they are to conduct data mining tasks on the joint data set that is the union of all individual data sets. Each wants the others to find as little as possible of their own private data. To focus the discussion on privacy-preserving collaboration, we will regularly use two parties Alice and Bob. We focus on vertically partitioned data [56] (Fig. 1).

Every record in the database is an attribute-value vector. Alice owns one part of that vector and Bob owns the other part. In the case of more than two parties, then every party will own some part (a number of attributes) from the attribute-value vector. Note that, for vertically partitioned data, the more parties are involved, the more attributes are involved and the higher the dimensions of the attribute-vectors. For simplicity, we can identify each attribute (column or field) with one party (so the dimension m of the records is also used as the number of

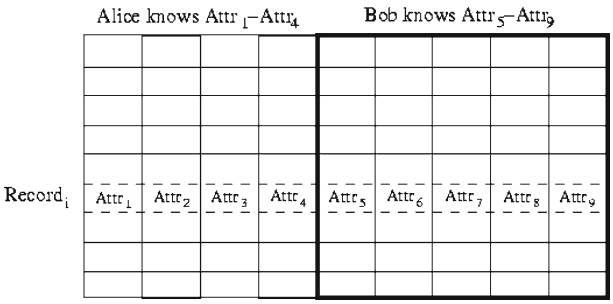

Fig. 1 Vertically partitioned data

parties). Obviously, there would be fewer parties than dimensions (for example, in Fig. 1 the two parties hold 9D records). However, we consider Alice as four virtual parties (one for each of the columns) and Bob as five virtual parties each controlling one of Bob's columns. This simplifies the notation in the algorithms (and communication between two virtual parties of the same party does not need to occur).

A direct use of data mining algorithms on the union of the data requires one party to receive data (every record) from all other parties, or all parties to send their data to a trusted central place. The recipient of the data would conduct the computation in the resulting union. This naive solution is unacceptable from the privacy perspective. However, the cost of this distributed non-private solution (*DNPS*) has been used in the past as a benchmark for evaluating the overhead required for privacy.

Our approach will make reference to the theory developed under the name of "secure multi-party computation" (SMC) [27]. Yao's millionaires problem [63] provides the origin for SMC. In this problem, Alice holds a number a while Bob holds b. They want to identify who holds the larger value (they compute if $a > b$) without either learning anything else about the other's value.

Secure multi-party computation under the semi-honest model [27] has regularly been used for privacy-preserving data mining [17,19,26,53]. Here we use as the fundamental point of reference the semi-honest model as well, which means all parties will follow the protocol since all are interested in the results. However, all parties can use all the information collected during the protocol to attempt to discover information on the private data or some private values from another party. We accept that any information that can be inferred from the privately computed output and the inputs of one party, is acceptable for that party to discover.[2]

[2] The formal definition of the semi-honest model for two parties [27, Defintion 7.2.1, Page 620] is slightly different than for three or more parties [27, Definition 7.5.1, Page 696]. For three or more parties one most additionally prove that any proper subset of the parties colluding among themselves do not learn any additional information about the private data of those that remain honest. The data mining community has generally ignored this technical difference between the two-party case and settings with three or more parties by assuming there will not be a case where a subset of the parties collude. For example, the semi-honest model is illustrated to the data mining community with the protocol SECURE SUM for $m > 2$ parties (see Protocol 4 later) in a text [56] that admits no party can collude with the originator of the protocol or the protocol fails. The same text suggests [56] an extension that suffices for an honest majority (so it is formally insecure for a honest minority). Another example is the PRIVATE GENERALIZED SCALAR PROTOCOL—PROTOCOL 4 [26, Page 119] which improves the security of at least two other protocols for computing the scalar product. Again, the authors admit that "when Alice colludes with other parties, then privacy can be compromised". In other cases, the authors express a security result as a theorem for the semi-honest model without any collusions [35,57]. Therefore, as our formal model, we will follow the community's practice, and use the semi-honest

The current k-NN privacy-preserving algorithms at some stage make use of the theoretical generic "secret shares" result for computation with data split across several parties. The SMC literature has a general solution for all polynomially bound computations [28]. This generic "secret shares" solution computes $f(\vec{x}, \vec{y})$ for a polynomial-time f using private input $\vec{x}$ from Alice and private input $\vec{y}$ from Bob. Alice learns nothing about $\vec{y}$ except what can be computed from $f(\vec{x}, \vec{y})$ and similarly Bob learns nothing about $\vec{x}$ except what can be inferred from $f(\vec{x}, \vec{y})$ and $\vec{y}$. Why, if such a solution exists, is there so much interest in protocols for SMC? The first consideration is that the general solution requires f to be explicitly represented as a Boolean circuit of polynomial size. Second, even if represented as a circuit of polynomial size in its input, the input must be very small for the construction of the circuit to be practical. This means, the sub-task that uses this result must be on small inputs, a constraint difficult to meet in data mining applications. Third, the constants involved are not small. The circuit must be described for each input size n; once the circuit is described the parties enter into a protocol holding shares of the inputs to gates and shares of the outputs of gates. Fourth, the literature shows that much more efficient solutions exist for special cases of f. In other cases, researchers are prepared to describe pragmatic solutions that reveal some information that can be considered innocuous. Also, the usage of a circuit-based protocol as a subroutine in another protocol enables construction of more complex and secure protocols, but transmits the impracticality of the generic "secret shares" result further.

2.1 Yao's two millionaires problem—solution with shares

One advantage of the "secret shares" theoretical result is that one can 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 us to use a protocol for one task (like Yao-comparison of two values) in a larger protocol (e.g. sorting).

We present here a solution to Yao's millionaires problem, that 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

$$r_a + r_b = \begin{cases} 1 & \text{if } a > b, \\ 0 & \text{if } a < b. \end{cases}$$

There are several privacy-preserving data-mining algorithms [35,47,54] that invoke a subroutine for Yao-comparison with shares, and all of them rely on the (circuit evaluation) generic "shares" theoretical solution by Goldreich [27].[3] Hence, they seem laborious for implementation. We present here a practical and inexpensive solution that we apply in our algorithms. This solution can also alleviate the implementation issues for the above-mentioned protocols. It will be close to an ideal solution in the semi-honest model. It uses a third untrusted party[4] (also commonly used in the privacy-preserving literature [19]). It is also common that when there are more than two parties, they take turns performing the role of the third party for two others [53]. Even for fundamental protocols, like oblivious transfer, there has been an inter-

Footnote 2 continued
model where the definitions between two parties and three or more parties are equivalent by assuming parties do not collude (we will indicate the threats of collusion when appropriate).

[3] Within the security community, debate remains lively about requirements (and models) for this problem [44], but we frame our discussion under the semi-honest model.

[4] This may seem a strong assumption, but is not rare in reality. An on-line auction is an example of such a party, because buyers and sellers assume that the auctioneer is non-colluding.

est in using a third party [45]. In some protocols, like unconditionally secure commitment schemes, the third party is absolutely necessary for perfect concealment [11].

Clearly, 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 [13]). Such a party only assists in performing the calculations. By definition, this party does not collude with Bob, neither with Alice (and Alice and Bob would not collude since they are two millionaires that do not trust each other). Therefore, in this case, the semi-honest model with three or more parties is equivalent to semi-honest model with two parties.

Protocol 1 SIGN- BASED PROTOCOL with secret shares for Yao's millionaires problem.

1. The third party generates a random number R_a and sends R_a to Alice.
2. Alice generates a random number R, where $R \in \Re\backslash\{0\}$ and sends $(R, a \cdot R + R_a)$ to Bob.
3. Bob adds $-b \cdot R$, and sends $(a - b)R + R_a$ to the third party.
4. The third party subtracts R_a and checks whether $(a - b)R > 0$.
5. The third party generates two pairs of values (r_a^0, r_b^0) and (r_a^1, r_b^1), where $r_a^0 + r_b^0 \equiv 0 \bmod 2$ and $r_a^1 + r_b^1 \equiv 1 \bmod 2$ (r_a^0, r_b^0, r_a^1 and $r_b^1 \in \mathbb{Z}_2 = \{0, 1\}$ with r_a^0 and r_a^1 random bits and r_b^0 and r_b^1 bits determined by the congruence). 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^1) to Bob.
6. If $sign(R) = 1$ (i.e. $R > 0$) Alice and Bob use as their shares the random numbers received as the first value in the pair sent by the third party. If $sign(R) = -1$, each picks as their secret share the random number received as the second number in the pair received as the third party's message.

2.1.1 Illustration of Yao's two millionaires problem—solution with shares

Assume that Alice holds $a := 25$ and Bob holds $b := 37$. They want to know whether $a > b$ with shares.

1. The third party generates a random number $R_a := 45$ and sends R_a to Alice.
2. Alice generates random number $R := -14$, where $R \in \Re\backslash\{0\}$ and sends $(R, a \cdot R + R_a) = (-14, 25 \cdot (-14) + 45) = (-14, -305)$ to Bob.
3. Bob adds $-b \cdot R = -37 \cdot (-14)$, and sends $R(a - b) + R_a = -14(25 - 37) + 45 = 213$ to the third party.
4. The third party subtracts R_a and checks whether $R(a - b) > 0 \Rightarrow 213 - 45 > 0$.
5. The third party generates two pairs of values $(r_a^0, r_b^0) := (0, 0)$ and $(r_a^1, r_b^1) := (0, 1)$, where $r_a^0 + r_b^0 = 0$ and $r_a^1 + r_b^1 = 1$. Since $R(a - b) = 168 \geq 0$, so the third party sends $(0, 0)$ to Alice and $(1, 0)$ to Bob.
6. Because $sign(R) = -1$, each picks as their share the random number received as the second number in the message, that is 0 for Alice and 0 for Bob.

2.1.2 The security of the SIGN- BASED PROTOCOL *with shares*

Let us see what every party obtains from this protocol.

1. Alice obtains R, R_a and the set $\{r_a^1, r_a^0\}$. However, Alice has no way of telling which one is which in the set $\{r_a^1, r_a^0\}$.
2. Bob obtains R, the set $\{r_b^1, r_b^0\}$, and $(a)R + R_a$. Again, Bob can not tell which one is which on the set $\{r_b^1, r_b^0\}$.

3. The third party obtains R_a, (r_a^1), (r_a^0), (r_b^1), (r_b^0), and $(a-b)R$.

From the discussion above, the following result follows.

Theorem 1 *If $a \neq b$, the* SIGN- BASED PROTOCOL *with shares is secure within the semi-honest model of computation.*

We emphasize that in the SIGN- BASED PROTOCOL neither Alice nor Bob learn anything. Alice does not receive any messages nor values from Bob and what she receives from the third party is random numbers. Similarly, Bob receives $(R, (a)R + R_a)$; this is the only message from Alice and since R_a is a random number he cannot infer a. Neither can he infer the choice by the third party.

Some researchers use the tools by Goldreich [27] to establish the security of the protocol. In particular, a proof by simulation is sometimes performed there [54]. Here a proof of Theorem 1 by simulation is obvious.

Proof Bob just needs to get a random value for $(a)R + R_a$ and in polynomial time can add $(-b)R$. The output is given by the third party. Alice's simulation is even more trivial since she receives nothing from Bob, and the only output is also from the third party. Let $a \neq b$, the third party only receives $(a-b)R$ where R is a random number, thus it can also be simulated. □

We have chosen to define a Yao-comparison also when $a = b$. When values are equal, the predicate $a > b?$ receives the value *true* or *false* by considering the party that supplies the second argument as the holder of a larger value. This implements implicitly and effectively a comparison where a later indexed party is considered to have a larger value among parties with equal values (which is useful for computing the Chessboard distance later in Protocol 5).

2.1.3 The implementation of the sign-based solution with shares

Theorem 1 proves the ideal protocol is secure, but as presented it uses real numbers and cannot be considered efficient. Implementation requires some adaptation, as for example, 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). Alice and Bob will map their values from whatever total order by a strictly monotonic function to $[0, M]$. 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.

The implementation will not reveal any information about Bob's value to Alice, since nothing that Alice receives depends on Bob's b except the output of the protocol. The implementation faces two challenges, namely limiting what Bob may learn about Alice's value and limiting what the third party learns on Alice's value, Bob's value or both. We address first the second case.

Note that the third party does not know the value $a - b$, 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 inaccessible to the third party). Theorem 1 does not cover this case.

Risk that the third party learns $|a-b|$. The trusted party does learn $(a-b)R$ and because $|R|$ is bounded (in fact, the most conservative assumption is that the distribution of $|R|$ will be known 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. Thus, the implementation of our protocol no longer satisfies Theorem 1 for the third party.

Risk that the third party learns something about a (and thus something about b). If we consider the values of $|a-b|$ given that $a, b \in [0, M]$, we can see that $|a-b| \in [0, M]$; however, the combinations that lead to a value are not equally many. For example, if $|a-b| = M$, then we know that $(a = 0 \wedge b = M) \vee (a = M \wedge b = 0)$. That is, there are only two possibilities. If $|a-b| = M-1$, then we know that there are only 4 possibilities (this are $a = 0 \wedge b = M-1$, $a = 1 \wedge b = M$, $a = M-1 \wedge b = 0$, and $a = M \wedge b = 1$). However, for other values of $|a-b|$, we do not gain additional information. For example $|a-b| = \lfloor M/2 \rfloor$ can be produced with any value of $a \in [0, M]$. Therefore, when the third party obtains the value $R(a-b)$, if the particular value of R has a few possibilities (for example, it is the maximum possible value) and $|a-b|$ is one of those extreme values with few possibilities, then the third party may reduce the universe of possibilities for $|a|$ (and therefore for $|b|$).

We evaluated the possibility of the third party bounding the range for $|a-b|$ and found the third party would still have large amounts of noise. We conducted experiments where we kept $|a-b|$ constant. For each value of $|a-b|$ at least 250 executions of the SIGN-BASED PROTOCOL were executed. In each, the third party used the mechanism above to estimate $|a-b|$. The range of values for which the estimate of $|a-b|$ was explored was 1 to 4,000. Figure 2 shows that with 95% confidence the expected relative error is around 50%, and the maximum relative error is over 99%. The plot of relative error zooms into the range $|a-b| \in [1, 1{,}000]$ to show that even with small values, the relative error remains essentially constant. We regard this leak of information in the SIGN- BASED PROTOCOL as innocuous given the estimates of $|a-b|$ have at least 50% expected relative error.

Secondly, the risk of a leak because of extreme values is very low already. If M is at least 255 and $R \in [256, 1{,}024]$, then the value $|a-b| = M$ would happen with probability $2/255^2$ and R would be largest with probability $1/768$, thus the chance of the SIGN- BASED PROTOCOL implementation running into these values is less than 2 in 10 million.

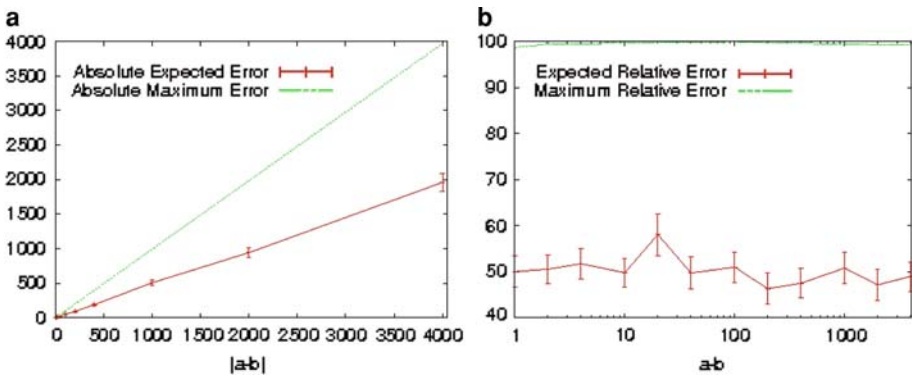

Fig. 2 Expected absolute and relative error (with 95% confidence intervals) when the third party estimates $|a-b|$, as well as maximum absolute error and maximum relative error over 250 execution of the SIGN- BASED PROTOCOL

Nevertheless, we provide a further enhancement that will result in an efficient implementation, but it will not be the of the SIGN- BASED PROTOCOL as presented earlier. Theorem 1 will not hold for the implementation, but we will introduce other formal aspects regarding its security. The implementation will be parameterized so that it is possible to arbitrarily increase the uncertainty in the third party. To hide information on $|a-b|$ from the third party, Alice and Bob will perform an additional number of tests. Here Alice and Bob use dummy values a and b, some of which have $a = b$. Only Alice and Bob know the index of the test that actually corresponds to the comparison of their private values.

Protocol 2 ITERATION OF COMPARISON- TESTS

1. Alice and Bob (without the third party's involvement), agree on a integer parameter r (or repetitions), a random uniformly distributed integer $i \in [1, r]$.
2. For each value $j \in [1, r]\backslash\{i\}$, Alice flips a coin with probability $1/2$ and if heads, Alice adds j to a set of e. The set e is initially empty and after the above step is a set of different integers in $[1, r]$ (of dummy tests where Alice and Bob will supply each equal values).
3. Alice and Bob agree on random values $c_1, \ldots, c_{\|e\|}$ that are in the range of a and b.
4. Alice and Bob run the SIGN- BASED PROTOCOL r times, providing dummy values except for the ith test where Alice provides a and Bob provides b. If the jth dummy test has its index in e, they both provide the value c_j.

In our experiments with real databases we found that equal distance values are extremely rare in some databases and common in others. Thus, handling this issue properly is essential, but we will not emphasize it much further (implementations for settings were a and b are of a small enumerated type, for example Boolean values, will use more rounds of the ITERATION OF COMPARISON- TESTS with more dummy tests).

While the ITERATION OF COMPARISON- TESTS is formally not compliant with the semi-honest model because the third party knows that one of these tests gives some idea of the range of $|a-b|$, the certainty on any particular value decreases monotonically with the number of iterations.[5] An alternative model for assessing the privacy of protocols in privacy-preserving data mining is the *weak* model. The weak model was used for many algorithms involving matrix operations and in particular, linear regression [18,56]. In this model, security is regarded with respect to certainty. Therefore, one party is considered to not have breached the security as long as there are an infinite number of possibilities for the values of the other parties. This model has been criticized, and in many cases rejected, because it can consider secure a protocol where one party learns significant information about another party's data. For example, Alice could learn that Bob's b value is in a small range. While there are an infinite number of rationals (or reals) in this interval, this could provide enough precision for it to be considered a security leak. Learning or discovering an interval is discovering a distribution of the value. If the distribution has very small variance, although a large range, the security leak could be serious.

While some protocols for vector and matrix operations have been dismissed as only secure on the weak-model and not in the semi-honest model, we believe one cannot discard the merit of these protocols; particularly if they are regarded as not secure in the semi-honest model by a technicality. Case in point is the protocol for scalar product that provides the output in shares [19]. This protocol is regarded as secure in the weak-model sense because it requires a commodity server. We argue here that the commodity server does not contradict the spirit

[5] If we are in a situation where $m > 3$ not colluding parties are involved, with $P_3, \ldots P_m$ as commodity third parties, alternating to help P_1 and P_2 perform a SIGN- BASED PROTOCOL, then each party P_i ($\imath \in \{3, \ldots, m\}$) is not even sure that one of the test is not using dummy values.

of the semi-honest model. We can consider the commodity server as a third party in the protocol, with empty input and empty output. Then, the three parties would be interested in computing $f(\vec{x}, \vec{y}, \lambda) = s_A + s_B$, where the input λ of the third party (the commodity server) is empty and will not affect the output value $s_A + s_B$ discovered by Alice and Bob with respected private shares. As we mentioned before, many times a protocol in the semi-honest model among more than two parties requires a sub-protocol in which two parties compute a value with the assistance of a third. We hope that the above discussion makes clear that our iterated SIGN- BASED PROTOCOL is not only secure on the *weak model* but is arbitrarily close to being secure in the semi-honest model.

We make this last statement formal with the notions of *perfect secrecy*[6] and of *statistical secrecy*.[7] We now present the specific definitions of perfect security and statistical security we will use.[8] Consider an encryption scheme $ES(\mathcal{K}, E_k, D_k)$, where $\mathcal{K}$ is a random variable (representing the distribution of the keys), E_k is the family of encryption functions (which can possibly be randomized) and D_k is the decoding function. That is, $D_k(E_k(p)) = p$ for all possible plain texts p.

Definition 2.1 The encryption scheme has perfect security if for any two plain-text instances p_1 and p_2 and a cipher-text c, we have

$$Pr[E_k(p_1) = c] = Pr[E_k(p_2) = c],$$

where probability is taken over the distribution of $k \in \mathcal{K}$.

For example, consider how well are Alice and Bob hiding the secret "ith round is the actual round to compare their values". It is not hard to see that for any set $T \subset [1, r]$, we have $Pr[\text{actual round} \in T] = \|T\|/r$. This proves the following result.[9]

Theorem 2 *Protocol 2* ITERATION OF COMPARISON- TESTS*, hides which round is the round involving the actual values a and b form Alice and Bob with perfect secrecy.*

However, it is intuitively clear that when $a = b$, there would be more rounds where the third party observes a Yao-comparison with $a = b$ than if $a \neq b$. Thus, the protocol is not perfectly secure in this regard, but we will show it is statistically secure and parameterized to achieve any level of statistical security. First, we must recall the definition of statistical distance.

Definition 2.2 Let X and Y be two distributions over $\{0, 1\}^t$. The statistical distance between X and Y, denoted $\Delta(\mathrm{X}, \mathrm{Y})$ is

$$\max_{T \subset \{0,1\}^t} |Pr[\mathrm{X} \in T] - Pr[\mathrm{Y} \in T]|.$$

Moreover, if $\Delta(\mathrm{X}, \mathrm{Y}) \leq \epsilon$, then we say the distributions are ϵ-equivalent and write $\mathrm{X} \equiv_\epsilon \mathrm{Y}$.

[6] Shannon seminal paper defined [48] " 'Perfect Secrecy 'is defined by requiring of a system that after a cryptogram is intercepted by the enemy the a posteriori probability of this cryptogram representing various messages be identically the same as the a priori probabilities of the same messages before the interception".

[7] Informally, perfect secrecy means that an unrestricted adversary gains absolutely no information about the secret, while for statistical secrecy, the adversary learns a little about the secret and this is measure by the difference in the distribution before and after the protocol [7,16].

[8] Our definition is equivalent to Definition 2.2 [49, Page 48] and Shannon's when we naturally assume that the a posteriori probability is the uniform probability.

[9] This is analogous to the proofs that *Shift Cipher* and that *One-Time Pad* have perfect secrecy [49, Chapter 2].

We are now in a position to define statistical secrecy. An encryption scheme is ϵ-secure if for any two plain-texts p_1 and p_2, the distributions $E_k(p_1)$ and $E_k(p_2)$ are ϵ-equivalent. We can now establish the following result.

Theorem 3 *Protocol 2* ITERATION OF COMPARISON-TESTS*, is a parameterized protocol by the number of rounds r that hides the secret whether $a = b$ with statistical security at any required security level.*

Proof The definition of statistically secure requires that whatever event T in the probability space, the probability of observing any ciphertext landing in T (that is, $E_k(p_1) = c \in T$) is within ϵ of the probability of observing any other ciphertext $E_k(p_2) = c \in T$. But the number $\|e\|$ of rounds in the protocol where equal inputs are provided by the parts is distributed as the Binomial distribution with $r - 1$ trials and probability $1/2$ when $a \neq b$, and shifted by one when $a = b$. But the Binomial distribution converges to the Normal distribution and for r sufficiently large, the number of rounds where the third party observes $a = b$ will have probability ϵ close to whether it is the case that the actual values are equal or whether they are not. □

Even more levels of statistical security for $|a - b|$ can be obtained by Alice and Bob engaging in a previous protocol that chooses randomly a constant number of monotonic[10] functions from $[0, M]$ to $[0, M]$ (without knowledge from the third party). In fact, they can also chose anti-monotonic[11] functions. In this way, they can change the magnitude of $|a - b|$ to $|f(a) - f(b)|$ with a slight trade-off. Namely, because $[0,M]$ has $M+1$ values, there may be cases where $f(a) = f(b)$ but $a \neq b$. Nevertheless, the family of monotonic functions on $[0, M]$ is rich enough to select a subfamily to suit the balance between strict comparison and a randomized comparison that, with small probability, returns that $a = b$ rather than the true comparison of a and b.

Lets now look at the risk that Bob learns something about a. 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 trusted party chooses $|R_a| > M^2$. For example, Alice can chose R uniformly in $[-2M, M) \cup (M, 2M]$.[12]

However, Bob learns the value of R and the most conservative assumption is that Bob knows the distribution (typically uniform distribution) of R_a. Therefore, when Bob receives $a \cdot R + R_a$ and although R may be equally likely in its range $[R_l, R_h]$ and R_a is equally likely in it range $[R_{a,l}, R_{a,h}]$, the values of the form $a \cdot R + R_a$ are not equally likely (even if also a is equally likely in $[0, M]$). For illustration, we can consider $M = 10$, $R_a \in [R_{a,l} = 100, R_{a,h} = 1,000]$ and suppose $R = 21 > M$. The possible values that Alice may send to Bob have a distribution as per Fig. 3a.

For example, for the values $a \cdot R + R_a$ in $[101, 120] \cup [1{,}190, 1{,}210]$, there is only one value of a. That is, if Bob receives 1,192, he would be able to find that Alice's value is $a = 1$. Similarly, for the values $a \cdot R + R_a$ in $[121, 141] \cup [1{,}169, 1{,}189]$, there are only

[10] A function $f : [0, M] \to [0, M]$ is monotonic if for all $x, y \in [0, M]$ with $x < y$, we have $f(x) \leq f(y)$.

[11] A function $f : [0, M] \to [0, M]$ is anti-monotonic if for all $x, y \in [0, M]$ with $x < y$, we have $f(x) \geq f(y)$.

[12] Using the range $[-4M, M] \cup [M, 4M]$, or a mapping of this to $\mathbb{Z}_m$, for large enough m ensures no overflow. In fact, we could chose R uniformly in $[-K, -M] \cup [M, K]$, where K regulates the complexity of the protocol. For example, letting K be the largest integer representable in as many bits as function C of the number of bits of $|a| + |b|$, results in complexity regulated by C.

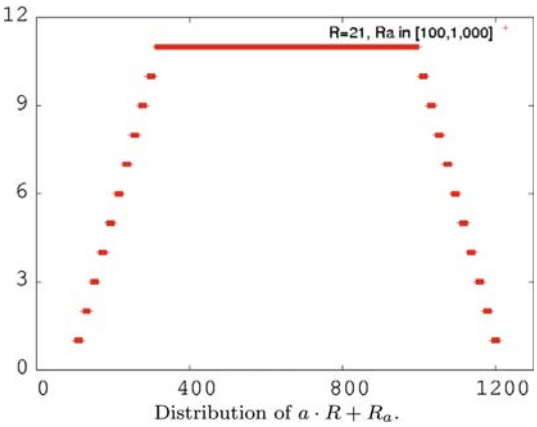

Fig. 3 Potential risk that Bob learns a smaller range for Alice's value a

two possible values for a (if Bob gets 1,189, he learns $a = 10$ or $a = 9$). However, in the range [310, 1,000], Bob remains uncertain among the 11 possible values for a.

Clearly, given R, analyzing the frequencies of the value $a \cdot R + R_a$ does not support one value of a more than it supports another one that can be expressed this way. For illustration, consider again the setting of the distribution in Fig. 3a. Knowing that 1,153 corresponds to three values of a does not favour any of these three values (although fewer than the original 11 possibilities for a, Bob will still be equally uncertain about $a = 10$, $a = 9$ or $a = 8$). Thus, direct implementation of our protocol no longer satisfies Theorem 1 for Bob.

However, there are far more values that leave Bob as uncertain (in the example above 691 values of the from $a \cdot R + R_a$ maintain the 11 possible values for a). More realistic values are M at least 255, (say 256) $R_a \in [R_{a,l} = 65,536, R_{a,h} = 16,777,216]$ and $R \in [256, 1{,}024]$. In this case, the number of values of the form $a \cdot R + R_a$ that reduce the $M + 1$ is no more than $2MR$. This represents less than 4% of the values Bob can see. Thus, most of the time Alice does not have to worry about what Bob will learn. However, if this is an issue, (for example, it is harder for Alice to hide the values $a = 0$ or $a = M$ than $a = \lfloor M/2 \rfloor$), Alice can actually request several values R_a from the commodity server and chose one that lands in $[R_{a,l} + RM, R_{a,h} - RM]$. Then, it lets the third party know which of the several values it selected without disclosing this to Bob.

Moreover, because Alice and the third party can set parameters of the protocol before Bob, we can produce a parameterized version of the protocol that can be made statistically secure for any desired level $\epsilon > 0$.

Theorem 4 *For every $\epsilon > 0$, there is a parameter s so that we can tune the implementation of the* SIGN- BASED PROTOCOL*, so that the protocol is statistically secure at level ϵ.*

Proof Recall that $[0, M]$ is the range of values for Alcie's a and Bob's b with M a constant. Given $\epsilon > 0$ chose s such that $2M/(2s + 1) < \epsilon$. Recall that Bob receives $(aR + R_a)$. We can consider this as the encryption scheme $E(a) = a + R_a/R$. Since an implementation will use floating point arithmetics, scaling by the floating-point unit, we can consider the random variable $\mathcal{K} = R_a/R$ has a range V of integer values given by $V = \{-s, -(s-1), \ldots, -1, 0, 1, \ldots, s-1, s\}$. Then, the third party and Alice agree so that

$$Prob[\mathcal{K} = x] = \begin{cases} 1/(2s+1) & \text{if } x \in V \\ 0 & \text{otherwise.} \end{cases}$$

That is, the probabilities are uniform in V. For simplicity of notation, for any value c, let $V + c$ denote the set $\{-s + c, -(s-1) + c, \ldots, -1 + c, c, 1 + c, \ldots, s - 1 + c, s + c\}$ of values in V shifted by c. Then, for any two integer values $a_0, a_1 \in [0, M]$ we have

$$\| (V + a_0 \cup V + a_1) \setminus (V + a_0 \cap V + a_1) \| \leq 2M.$$

Note that the probability for any event $T \subset \Re$, is the same as analyzing it for $T' = T \cap V$. Moreover,

$$Prob[E(a_0) \in T'] = \sum_{x \in T'} Prob[E(a_0) = x] = \|T'\|/(2s+1).$$

We are interested in an event T' that maximizes $|Pr[E(a_0) \in T'] - Pr[E(a_1) \in T']$. But if $t \in T'$ is such that $t' \in V + a_0 \cap V + a_1$, then $Pr[E(a_0) = t'] = Pr[E(a_1) = t']$. At the same time, if $t' \in V + a_0 \setminus V + a_1$ or $t' \in V + a_1 \setminus V + a_0$, then $|Pr[E(a_0) = t'] - Pr[E(a_1) = t']| = 1/(2s+1)$.

Therefore, the event T' that maximizes $|Pr[E(a_0) \in T'] - Pr[E(a_1) \in T']|$ is

$$T'_0 = (V + a_0 \cup V + a_1) \setminus (V + a_0 \cap V + a_1).$$

Moreover the maximum value is bounded by $\|T'_0\|/(2s+1) \leq 2M/(2s+1) < \epsilon$ as required. □

Protocol 3 IMPLEMENTABLE SIGN- BASED PROTOCOL with secret shares

1. Alice an the third party agree on distributions of R and R_a so that R/R_a has a range V so that $\|V \cap \mathbb{Z}\|$ has more than $2s+1$ consecutive integers, $2M/(2s+1) < \epsilon$, and the values in V have almost equal probability (while also $|R| > M$ and $|R_a| > M^2$).
2. They proceed with the SIGN- BASED PROTOCOL.

We have implemented the sign-based solution presented above in combination with Protocol 2. Note that for nominal or categorical types that are converted to ordinal types for comparisons (like Boolean values) the value of M is small and the choice of distributions for R and R_a is simple. The implementation trades-off uncertainty for the size of ranges in the implementation. This implementation minimizes exchanged messages and operations. Other implementations of SMC are recognized as expensive, originally directly implementing circuit evaluation [40], but recent efforts have made them feasible [43]. To the best of our knowledge, ours is the first implementation 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 [13] is linear on the number[13] of bits of $a + b$; however, it also requires a trusted party and very heavy cryptographic machinery. A solution that has been demonstrated to be efficient enough for ETHERNET networks [33] 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 [6] also require $O(log_2(a + b))$ rounds of oblivious transfer. Oblivious transfer implementations usually require at least two messages with a key each. The SIGN- BASED PROTOCOL requires three messages *in total* with size $\log_2(a + b)$ (one from the trusted party to Alice, one from Alice to Bob and one from Bob to the trusted party). In the last round, 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. The complexity analysis is so overwhelming clear in favor of the SIGN- BASED PROTOCOL that we feel direct comparison to

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

any other implementation of a solution to Yao's millionaires unnecessary. The SIGN- BASED PROTOCOL requires little machinery and thus it is also much easier to implement than any of the others. The fact that we may chose $|R_a| > M^2$ or even M^3 only adds liner complexity to the size of the very few messages our protocol requires. Note also that our protocol is connection-less, facilitating significantly the complexity of the networking machinery and also reducing other security risks. Note, however, that this also illustrates the power of the third party. Some of the other protocol mentioned here trade-off their independence from a third party by additional machinery.

3 Privacy-preserving metrics

One of the contributions of this paper is to show how to carry out the SMC computation of all *Minkowski* metrics (among them, the Euclidean metric) and also of the L_∞ distance. Also, we expect that in applications of vertically partitioned data, the fields (or columns) may be very diverse, including many units, and types, some being categorical and others ordinal or numerical. It is well known that in Instance-based Learning [60] or k-NN classification, typically the metric is a weighted convex combination of metrics for each attribute. The discussion of algorithms for *Minkowski* and L_∞ will enable algorithms for combinations of local metrics into a global metric. Equipped with this, we can show that data structures for associative queries are then readily suitable for privacy-preserving algorithms. The protocols in the previous section are involved in the computation of L_∞ metrics but not *Minkowski* distances.

3.1 *Minkowski* metrics

Obviously, if all parties know the rth *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 rth power. Conversely, if the parties find $[M(\vec{p}, \vec{q})]^r$, they can take rth roots and find the desired distance value. Since the ith party knows a range of the attributes of the vectors $\vec{p}$ and $\vec{q}$, it is not hard to see that

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

Letting v_i be the rth *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 rth *Minkowski* distance raised to the power r in the projection that they own). Protocols that compute distributed sums have received many names (for example, SECURE SUM [56]), so we reproduce here the necessary variant for clarity.

Protocol 4 THE *Minkowski* DISTANCE PROTOCOL: Add the m values among $m \geq 3$ parties.

1. The first party (Alice) generates a random number R and passes it to the mth party (this is like an XOR mask of random bits).
2. The mth party adds its value v_m^r to the random number R and passes the result to the $(m-1)$ party.

3. For $i = m - 1$ down to 2, the $(i-1)$th party adds v_i^r to the value received and passes the result to the $(i-1)$th party.
4. The protocol continues until Alice (the first party) gets $(v_2^r + \cdots + v_{m-1}^r + v_m^r + R)$, then she adds her value v_1^r and subtracts the random number R. She takes the rth root and announces the result to all parties.

Note that if we halt at step 3 with Bob (the second party), then we have a PROTOCOL FOR THE *Minkowski* DISTANCE WITH SHARED VALUES 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 $[M(\vec{p}, \vec{q})]^r = a + b$. In some applications (in particular, k-NN queries) the calculation of the metric is an intermediate step. Although the disclosure of distance values may be considered the release of innocuous information in some cases, 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 $s_a + s_b = [M(\vec{p}, \vec{q})]^r \mod F$. In a k-NN query, only the ids of the k records is the ideal answer and it is preferable than all the parties learning some exact values of the metrics to the query point.

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. In the core operations for the *SASH* (or similar data structures) rather than being interested in $dist(\vec{p}, \vec{q})$ itself, the question is whether "$dist(\vec{p}, \vec{q}) < dist(\vec{p}, \vec{r})$?" In the case of the *Minkowski* distance, this is also equivalent to asking "is $[M(\vec{p}, \vec{q})]^r - [M(\vec{p}, \vec{r})]^r < 0$?". For additional privacy, then, it is better if each party contributes the difference of its projections. That is, let v_i^r be the *Minkowski* distance (raised to the rth degree) between $\vec{p}$ and $\vec{q}$ in the projection owned by the ith party, and let u_i^r be the *Minkowski* distance of rth degree between $\vec{p}$ and $\vec{r}$ in the projection owned by the i-party. Then, to answer the question we compute the sum of the m values $(v_i^r - u_i^r)$ owned distributively by the m parties, and each party then can check where the sum stands relative to zero. SMC of the Euclidean distance is achieved by the above *Minkowski* algorithm in the case $r = 2$. Note also that if we consider the "shares" version of the protocol and $dist(\vec{p}, \vec{q}) = s_a + s_b$ and $dist(\vec{p}, \vec{r}) = s'_a + s'_b$ (with s'_a and s_a known by Alice and s_b, s'_b by Bob), we can still ask "$dist(\vec{p}, \vec{q}) < dist(\vec{p}, \vec{r})$?" as a Yao-comparison as $s_a + s_b < s'_a + s'_b$? is also $(s_a - s'_a) + (s_b - s'_b) < 0$? with $(s_a - s'_a)$ known to Alice and $(s_b - s'_b)$ known to Bob.

Theorem 5 *If three or more parties use* THE *Minkowski* DISTANCE PROTOCOL, *no party learns other parties' private data represented as an attribute in the feature vector. If the protocol is the shares distance version, even with two parties, no party learns any information.*

Proof The protocol calculates the distance between $\vec{p} = (p_1, \ldots, p_m)$ and $\vec{q} = (q_1, \ldots, q_m)$. During the calculation each party P_l $(l = 2, \ldots, m)$ obtains

$$S_l = (p_{l+1} - q_{l+1})^r + \cdots + (p_m - q_m)^r + (p_1 - q_1)^r + R$$

where R is a random number produced by the 1^{st} party. Thus, because R is random, 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 and taking rth 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 $i = 2, \ldots, m$.

The case for shares follows by the discussion above. □

Note that in the formulation of the theorem we have not mentioned the semi-honest model. However, this protocol is considered secure among the data mining community [56] because

each party can be simulated individually in polynomial time from its inputs and outputs and a random oracle assuming no proper subset of the parties colludes.

3.2 SMC chessboard or L_∞ distance

While *Minkowski* metrics combine the discrepancy in each attribute with a sum there is also the alternative of selecting the maximum difference as the overall metric. This leads to the L_∞ metric (some researchers prefer the name "Chessboard distance" which is defined as $dist(\vec{x}, \vec{y}) = \max(|v_1^x - v_1^y|, \ldots, |v_m^x - v_m^y|)$ where $\vec{x} = (v_1^x, \ldots, v_m^x)^T$ and $\vec{y} = (v_1^y, \ldots, v_m^y)^T$ are two vectors).

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 (thus, from now, we assume party i holds $|v_i^x - v_i^y|$ and the vectors have dimension m).

A first approach can use a version of Yao's protocol (without shares) as a subroutine to deploy a finding maximum algorithm based on $m - 1$ comparisons. For example, for $i = 1$ to $m - 1$ compare the maximum found in the ith first parties with the $(i + 1)$ party. While this works well, the $(i + 1)$th party must interact using the SIGN- BASED PROTOCOL with the holder of the maximum among the first i parties (thus, learning who holds the maximum so far and has won some comparisons). So this approach is not secure in the ideal sense of the semi-honest SMC since additional information besides the maximum among all m entries is leaked. Again, some may consider this information leak innocuous, and in that case, parties may use this proposed approach. However, we now present an approach that with some additional machinery is secure and practical. The additional machinery is how to compare two numbers and distribute the output into shares (Sect. 2.1).

Protocol 5 FIND MAXIMUM VALUE with shares.
The protocol starts with each party comparing its value to every other party. Note here that, the <Alice vs Bob> comparison is not the same as the <Bob vs Alice> comparison. For instance, if Alice compares with Bob and it happens to be that Alice's value is smaller than Bob's, then they will have shares C_{AB}^A and C_{AB}^B, where $C_{AB}^A + C_{AB}^B = 0$,[14] but if Bob compares with Alice the shares should add up to one. However, we do not need to compare again Bob's number with Alice in order to have shares C_{BA}^A and C_{BA}^B, where $C_{AB}^A + C_{AB}^B = 1$, since they are already provided by the third party in our SIGN- BASED PROTOCOL (see the Sect. 2.1). Thus, we will use it as the shares for the ⟨Bob vs Alice⟩ comparison.

1. Alice (the first party) compares her value with all others, then sums up her parts of the shares and puts it as the first component in her shares vector. All other parties put their shares, which come from comparisons with Alice, again as the first component of their shares vector.
2. Bob (the second party) compares his value with all others,[15] then sums up his parts of the shares and puts this sum as the second component in his shares vector. All other parties put their shares that come from these comparisons as the second component of their shares vector.
3. The protocol continues until each party's value will be compared with all others.

[14] Here the subscript AB means the ⟨Alice vs Bob⟩ comparison, whereas subscript BA means the ⟨Bob vs Alice⟩ comparison.

[15] Note that Bob does not need to compare with Alice again. He rather uses the other share provided from the third party.

Table 1 Shares after all comparisons

P_1	P_2	$\cdots$	P_m
$\sum_{j=2,\ldots,m} C^1_{1j}$	C^2_{12}	$\cdots$	C^m_{1m}
C^1_{21}	$\sum_{j=1,\ldots,m}^{j\neq 2} C^2_{2j}$	$\cdots$	C^m_{2m}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
C^1_{m1}	C^2_{m2}	$\vdots$	$\sum_{j=1,\ldots,m-1} C^m_{mj}$

This provides the information shown in the Table 1, where C^i_{ij} belongs to P_i, C^j_{ij} belongs to P_j, and

$$C^i_{ij} + C^j_{ij} = \begin{cases} 1 & \text{if } v_i > v_j, \\ 0 & \text{if } v_i \leq v_j. \end{cases}$$

Note that now, each column is owned by one party only; therefore we can treat them as the separate vectors distributed to each party. Moreover, for each party P_i, the sum of the elements in the ith row is $\sum_{j=1,\ldots,m}^{j\neq i} C^i_{ij} + \sum_{j=1,\ldots,m}^{j\neq i} C^j_{ij}$, which will show us exactly how many v_j were smaller than v_i. The problem now reduces to finding the id of the maximum value in a sum of vectors.[16] This can be performed by SMC with 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.[17] 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. The two parties then will hold values so that $s_p + s = M \mod F$.

Theorem 6 *Whenever a secure Yao algorithm with shares is used for its comparisons. Protocol 5 is secure (in the semi-honest model).*

That is, in theory, Protocol 5 will fit the semi-honest model where each party can be simulated, because parties do not learn who holds the larger value in each comparison since they are all encoded in distributed shares. In practice, we would use or SIGN- BASED PROTOCOL and the mechanisms discussed earlier (Protocol 2). Together, these protocols ensure the information learned by a helper party about the party declared to have a larger value (in case projections of distances are equal and one is later in the ranking of parties) is extremely small.

[16] 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. The use of circuit evaluation for a Yao-comparison with shares faces the additional complexity of handling the case when the millionaires hold equal values. We believe solutions that resort to adding a second key (like the index of the vector [47]) to split up ties lead to larger circuits and more impractical solutions.

[17] Other alternatives for computing the maximum values in a sum of vectors under the semi-honest model provided no subset of the parties colludes also exist [21,53] but are computationally more expensive.

Table 2 Shares after all comparisons

Alice (1)	Bob (28)	Charles (12)	Daniel (6)
$C_{12}^1 + C_{13}^1 + C_{14}^1$	C_{12}^2	C_{13}^3	C_{14}^4
C_{21}^1	$C_{21}^2 + C_{23}^2 + C_{24}^2$	C_{23}^3	C_{24}^4
C_{31}^1	C_{32}^2	$C_{31}^3 + C_{32}^3 + C_{34}^3$	C_{34}^4
C_{41}^1	C_{42}^2	C_{43}^3	$C_{41}^4 + C_{12}^4 + C_{43}^4$

Table 3 Shares after all comparisons

Alice (1)	Bob (28)	Charles (12)	Daniel (6)
$0.2 + 1 + 12$	-0.2	-1	-12
0.7	$0.3 + 13 + 1.1$	-12	-0.1
0.5	-1	$0.5 + 1 + 0.4$	0.6
22	1.8	3	$-21 - 1.8 - 3$

3.2.1 Illustration of SMC Chessboard of L^∞ distance calculation

Suppose there are given $\vec{x}^T = (12, -23, 5, 8)$ and $\vec{y}^T = (13, 5, -7, 13)$, and assumed these are owned by four parties (Alice, Bob, Charles and Daniel) each holding one attribute. Since 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 4 parties has the largest value. Thus, at present Alice holds $|12 - 13| = 1$, Bob holds $|-23 - 5| = 28$, Charles holds $|5 - (-7)| = 12$ and Daniel holds $|8 - 13| = 6$.

The protocol starts with each party comparing its value to every other party. This provides the information in Table 2, where C_{ij}^i is the share that P_i holds, C_{ij}^j is the share that P_j holds, and

$$C_{ij}^i + C_{ij}^j = \begin{cases} 1 & \text{if } v_i > v_j, \\ 0 & \text{if } v_i \leq v_j. \end{cases}$$

Assume the C_{kj}^i are as in Table 3. If we sum up all the components[18] in each row we will obtain

$$\begin{pmatrix} 0 \\ 3 \\ 2 \\ 1 \end{pmatrix}.$$

Since the ID of the maximum is 2, the winner is Bob, the second party.

[18] This can be performed by SMC with the MAXIMUM VALUE IN THE SUM OF VECTORS protocol [2].

3.3 Combinations of metrics

Protocol 5 for the Chessboard distance and Protocol 4 with shares illustrated with the *Minkowski* metric are 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 the party 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 our Protocol 4 as we illustrated with the *Minkowski* distance (with both versions, with shares or disclosing the global metric value). Alternatively, the global metric could be a weighted maximum $\max_{i=1}^{m} \omega_i v_i$. In this case, our Protocol 5 (previously illustrated with the Chessboard metric) 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.

Note however, that, if the global metric is a sum of local metrics, one may be tempted to parallelize Protocol 4. Then Alice would add one random vector $\vec{R}^a = (R_a, \ldots, R_a)$ to its projection of all the metrics, and pass it to the m-party, which would add its projection and pass the vector down to the $(m-1)$th party, and so on. This does not represent a real saving except using one rather than n random values generated by Alice, but it allows each party to learn the distribution of the projection of the distance values for the previous parties in the line-up.

3.4 Other metrics

Other metrics common for large records (for example, between Web visitation paths [22]) are very important for high dimensional settings. The first one of these metrics is the Hamming distance H. Here $H(\vec{p}, \vec{q})$ is the number of entries where the vectors $\vec{p}$ and $\vec{q}$ differ. One realizes that for vertically partitioned data, secure multi-party computation of this reduces to computing again the sum of the Hamming distance 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 $\vec{q}$. The dot (scalar) product $\vec{p}^T \cdot \vec{q}$ is, again, the sum of values that correspond to the dot-product in the local projection of each party, and we have already indicated how to perform Euclidean distances like $\|\vec{p}\|_2$. However, while the value

$$\cos(\alpha) = \frac{\vec{p}^T \cdot \vec{q}}{\|\vec{p}\|_2 \|\vec{q}\|_2}$$

can be computed securely by our earlier protocols, we now show that it also can be computed securely and split in shares $s_a + s_b = \cos(\alpha)$ where again, s_a is known by Alice only and s_b is known by Bob only. This will pave the way for using this metric in our associative query algorithms later. Since dot products on vertically partitioned data are sums and can be computed with shares by our Protocol 4 we have constants A_1, A_2 and A_3 known only by Alice and B_1, B_2 and B_3 only known by Bob so that

$$\begin{aligned} \cos(\alpha) &= \frac{A_1 + B_1}{(A_2 + B_2)(A_3 + B_3)} \\ &= \frac{A_1 + B_1}{A_2 A_3 + A_2 B_3 + B_2 A_3 + B_2 B_3} \end{aligned}$$

$$= \frac{A_1 + B_1}{A_2 A_3 + (A_2, A_3)^T \cdot \begin{pmatrix} B_3 \\ B_2 \end{pmatrix} + B_2 B_3}.$$

Using the SCALAR PRODUCT [19] with shares,[19] 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 $B_5 = B_4 + B_2 B_3$ and $A_5 = A_2 A_3 + A_4$ we have the derivation.

$$\cos(\alpha) = \frac{A_1 + B_1}{\underline{A_2 A_3} + \underline{A_4} + \underline{B_4} + \underline{B_2 B_3}} = \frac{A_1 + B_1}{A_5 + B_5}$$

where A_5 is only known by Alice and B_5 is known by Bob.

This last division could be computed securely by applying a SECURE DIVISION PROTOCOL [17]; however, this does not result in a metric split on additive shares useful for k-NN queries. The next subsection handles this.

3.4.1 New division protocol with secret shares

The most common SMC DIVISION PROTOCOL [17], does not provide an answer with shares, it gives an answer to one party only.

Protocol 6 In the division protocol, Alice holds (a_1, a_2) and Bob holds (b_1, b_2), the goal for Alice is to obtain A and for Bob to obtain B, where

$$A + B = \frac{a_1 + b_1}{a_2 + b_2}.$$

1. Alice produces random number r_1 and Bob produces random number r_2.
2. Using the SCALAR PRODUCT PROTOCOL ([17] or [19][20]) which provides an answer to the one party only, Alice can obtain

$$r_2(a_2 + b_2) = (a_2, 1)^T \cdot \begin{pmatrix} r_2 \\ r_2 b_2 \end{pmatrix}$$

   and Bob can obtain

$$r_1(a_2 + b_2) = (b_2, 1)^T \cdot \begin{pmatrix} r_1 \\ r_1 a_2 \end{pmatrix}.$$

3. Alice and Bob again perform the SCALAR PRODUCT [19,26], which provides an answer with shares A and B, using the input vectors from each party, because

$$A + B = \left(r_1 a_1, \frac{1}{r_2(a_2 + b_2)} \right)^T \cdot \begin{pmatrix} \frac{1}{r_1(a_2 + b_2)} \\ r_2 b_2 \end{pmatrix}$$
$$= \frac{r_1 a_1}{r_1(a_2 + b_2)} + \frac{r_2 b_2}{r_2(a_2 + b_2)} = \frac{a_1 + b_1}{a_2 + b_2}.$$

[19] SMC of scalar product protocols has been scrutinized extensively, recently two protocols [17,52] have been shown [26] to be vulnerable to attacks in the case of binary entries and with low frequency, but an improved alternative (with answers in shares) has also been proposed [26].

[20] Here Bob sets his private share $V_2 = 0$. As the authors of this protocol remarked in the original paper [19](page 6), this does not let Alice to learn any of Bob's private data.

Clearly, all these protocols for alternative metrics are secure in the semi-honest model as commonly applied among the data community (i.e., when more than three parties, we assume no subset colludes).

3.5 Private k-nearest neighbours

We are now in a position to describe our first algorithm for k-NN queries. Given a set of vectors

$$\vec{v}_1 = (v_{11}, \ldots, v_{1m}), \ldots, \vec{v}_n = (v_{n1}, v_{n2}, \ldots, v_{nm})$$

together with a vector $\vec{q} = (q_1, q_2, \ldots, q_m)$ (where m is the number of parties involved in the computation), we must find $P(\vec{q}, k)$ where $P(\vec{q}, k)$ is the ids for the k nearest neighbours to the vector $\vec{q}$.

Protocol 7 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 $s_i^a + s_i^b = dis(\vec{q}, \vec{v}_i)$ (in fact, it is possible to assume encryption modulo a field F; i.e. $s_i^a + s_i^b = dis(\vec{q}, \vec{v}_i)$ mod F).
2. Alice computes the matrix D^A whose ij entry is $|s_i^a - s_j^a|$, while Bob computes the matrix $(D_{ij}^B) = (|s_i^b - s_j^b|)$.
3. Alice and Bob engage in Yao-comparisons with shares for each respective entry of D^A and D^B. Let σ_{ij}^a be Alice's share of comparing D_{ij}^A with D_{ij}^B (Bob's share is σ_{ij}^b).
4. Let Alice compute the vector V^A whose ith entry is $\sum_{j=1}^n \sigma_{ij}^a$ while Bob's V^B is such that $V_i^B = \sum_{j=1}^n \sigma_{ij}^b$.
5. Alice and Bob use the secure ADD VECTORS PROTOCOL[21] 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 $\vec{q}$).

Since Alice only learns counts of $d_{pq}^a - d_{pr}^a < d_{pq}^b - d_{pr}^b$ comparisons (or from the proof mechanisms of the semi-honest model), it cannot be simulated just from its inputs and outputs. Note that, if we use a metric like *Minkowski* (where all parties satisfy the semi-honest model in that they can be individually simulated), then all parties besides Alice can be simulated (this follows from the "Composition Theorem [27][22]). If Step 5 in Protocol 7 is replaced by the 'finding the kth largest element' [55], then provided there are no collusions

[21] The technique was introduced for manipulation of vector operations as the "permutation protocol" [17] and is also known as the "permutation algorithm" [53]. This protocol can provide an output in secret shares, so will discuss this as well. In this protocol, Alice has a vector $\vec{x}$ while Bob has vector $\vec{y}$ and a permutation π. The goal is for Alice to obtain $\pi(\vec{x} + \vec{y})$; that is Alice obtains the sum $\vec{s}$ of the vectors in some sense. The entries are randomly permuted, so Alice cannot perform $\vec{s} - \vec{x}$ to find $\vec{y}$. Also, Bob is not to learn $\vec{x}$.

[22] The Composition Theorem for the semi-honest model [27, Theorem 7.5.7, page 702] is technically only valid if the larger protocol $f \circ g$ is composed of two protocols f and g each secure in the semi-honest model (where g could involve k_1 parties and f could involve k_2 parties). However, as we explained earlier, if one assumes no proper subset of all parties will collude, then $f \circ g$ is secure when f and g are secure, whenever no subset of the parties collude. This follows from inspection of the proof of the Composition Theorem [27, Page 702]. And thus, the data mining community uses SECURE SUM inside other protocols [56] (for example, k-means clustering for vertically partitioned data [53]) and the larger protocols is considered secure with the caveat that no subset ever colludes, even for one instance of a subprotocol.

of any subset of parties, Protocol 7 is secure under the semi-honest model. For this, we use the versions with shares over a field for our protocols that compute metrics, and then it is possible to use the binary search over a field for the top k-neighbours [55]. The field binary search requires Yao-comparisons with shares (which we have now made more practical). This prevents Alice from learning the distribution of the distance values at the expense of the $O(n \log |F|)$ Yao-comparisons. Moreover, because the data is vertically partitioned, and the algorithm is distributed, it is impossible for one party to repeat an associative query many times without the participation (or knowledge) of the other parties. This is an extra security aspect of our context. However, if we use our protocol for the L_∞ metric, and the parties participate as helpers in SIGN- BASED PROTOCOL computations, in theory, they could not be simulated, but in practice, by Protocol 2, they would learn innocuous information. Protocol 7 is quadratic on n; for our implementation, we preferred the following protocol. Although it reveals what we believe is innocuous information, it requires $O(n \log n)$ time.

Protocol 8 FAST PP k- NN.

1. Same as Protocol 7.
2. Bob (the second party) uses a random value R_b only know to him and adds the vector $\vec{R}_b = (R_b, \ldots, R_b)$ (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 $i = 1, \ldots, n$, 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 $\vec{q}$) (Fig. 4).

Theorem 7 *The* PP k- NN *protocol does not allow any party to learn other parties' private data represented as an attribute in the feature vector.*

Proof Clearly each party except Bob and Alice participate in the protocol with an input that looks random (under the theory of SMC, all these parties can be simulated by polynomial algorithms that use as inputs guessed values from an oracle). Bob also participates in the protocol with what can be random input until Alice passes to him π^{-1} of the ids that constitute the result. This is secure because in the semi-honest model, Bob can learn anything that can be inferred from the result. Alice learns the distribution of the distance values translated by a random number. No other information is disclosed. Alice cannot be simulated with a polynomial algorithm that uses guessed values from an oracle or the protocol would fail. However, from a pragmatic point of view, Alice cannot link any of the values she receives to any party (because she ignores π, neither any of the distance values becomes known because they have R_b added to them and only Bob knows R_b). □

4 The private *SASH* data structure

For a privacy-preserving *SASH*, we must ensure that it is possible to implement all ADT-Dictionary operations (CONSTRUCT, INSERT, DELETE, SEARCH, etc.) and that each party will hold enough information to learn the desired output while being unable to discover data from records of other parties. The *SASH* data structure considers a universe of n objects (not necessarily vectors) for which a similarity measure $dist(u, v)$ exists between any two objects u and v. A *SASH* is a directed edge-weighted graph with the following main properties.

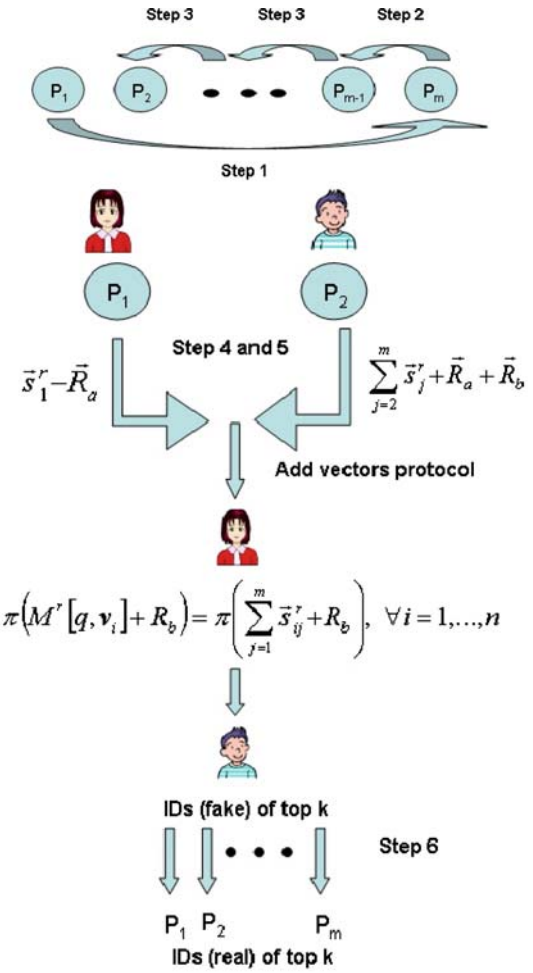

Fig. 4 k-NN calculation algorithm

- Each database object corresponds to a unique node. Little distinction will be made between a node, the object id or the database object to which it corresponds.
- The nodes are organized into a hierarchy of levels, ranging from a bottom level containing $\lfloor n/2 \rfloor$ nodes (the leaves), to a top level containing a single node (the root). With the possible exception of the bottom level, each level contains half as many nodes as the level below it, rounded down. The levels of the *SASH* are numbered from 1 to h.
- Edges within the *SASH* connect nodes from consecutive levels. Each node can have edges directed to at most p parent nodes as the level above it, and to at most c child nodes as the level below it. Every node except for the root must have at least one parent. The distance $dist(u, v)$ is always stored with each edge (u, v) at the time of its creation.
- Every node v (other that the root) has an edge directed to one parent $g(v)$ that is designated as its guarantor. The guarantor of v must have v as one of its children; v is called the dependant of $g(v)$. The requirement that every node have a guarantor ensures that every node is reachable from the root.

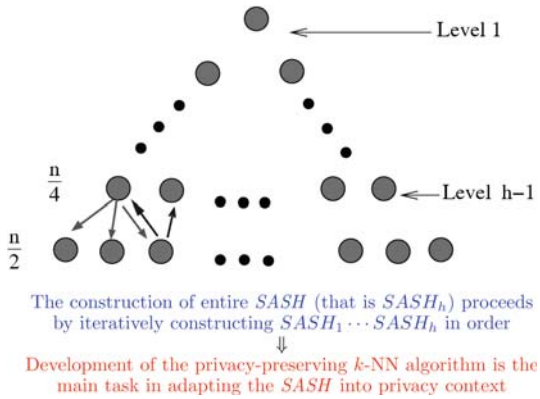

Fig. 5 Illustration of the strategy used in the *SASH* data structure

In our privacy-preserving *SASH*, all parties will know the entire graph structure of the *SASH*. That is, all parties will know how many nodes are at each level and what record corresponds to each node. Each party knows what are the parents and children of a node as well as who is its guarantor or the dependant of a node. For illustration, consider for a moment two parties with database objects as vectors in $2D$ and each coordinate known only by one party. Thus, Alice holds the first coordinate and Bob holds the second coordinate of each record. While both may know that, say, the fifth record in the database corresponds to the root of the *SASH*, neither will know the value of the other coordinate. The operations in the privacy-preserving *SASH* will inform all parties of the identifier of the record pointed by a node, its parents and its children, but will not reveal any values of the attribute-valued vector that describes the object. Distance values stored in edges are in fact distributed in shares (and not recomputed), this avoids the potential risk that repeated computation of a metric on the same vector may result in information leak to one party.

4.1 Constructing the *SASH*

The edges of the *SASH* heuristically minimise the distances between their endpoints. During the construction, each new node is attached to a small number of its near neighbours from the level above it. At the start of construction, the *SASH* is empty, and we insert all objects in a random and uniform order. We denote by $SASH_i$ the graph induced by the nodes from level 1 through i, for $1 \leq i \leq h$. Thus, $SASH_i$ is a *SASH* in itself. Iteratively constructing $SASH_1$, $SASH_2, \ldots, SASH_h$ results in the construction of the entire *SASH* (that is, $SASH_h$). The following algorithm shows how to construct $SASH_l$ given $SASH_{l-1}$ securely (for $1 \leq l \leq h$). This is where we add edges between nodes of the current last two levels (Fig. 5).

Algorithm privacy_Preserving_Connect_SASH_Level(l):

1. If $l = 2$, then every node of level 2 will have the root node as its sole parent and guarantor, and the root node will have all nodes of level 2 as its children and dependents. This completes the construction of $SASH_2$. Note that all parties must know the order of the data vectors (otherwise, the vertically partitioned data would not be able to join attributes for the same entity across parties). One can consider vertically partitioned data as several tables where a JOIN can be performed in a publicly known *id* attribute. Moreover, generating a random permutation of the object ids so that all parties know how to create the insertion order is no privacy risk. This would determine which entity is the root, what entities are in level two and all the edges between these two levels.

2. Otherwise, for the remaining steps, we have $l > 2$. For each node v of level l, choose using privacy-preserving protocol a set of up to p near neighbours $P_i(v, p)$ from among the nodes of each level $1 \leq i < l$. The algorithms for finding $P_i(v, p)$ uses $P_{i-1}(v, p)$, so we show how to compute $P_i(v, p)$ preserving privacy, assuming that $P_{i-1}(v, p)$ is computed preserving privacy (with a structural induction) as follows:
   (a) If $i = 1$, then $P_i(v, p)$ consists of a single node, the root (all parties know the id of the object here).
   (b) Otherwise, $i > 1$.
      i Let $P'_i(v)$ be the set of distinct children of the nodes of $P_{i-1}(v, p)$. All parties can obtain this, because once $P_{i-1}(v, p)$ is computed securely, its children can be found using the shared knowledge of edges linking ids.
      ii We compute $P_i(v, p)$ as the p nodes of $P'_i(v)$ closest to v, according to the measure $dist$ (we use the PP-k-NN in the Sect. 3.5). If $|P'_i(v)| < p$, then set $P_i(v, p) = P'_i(v)$.
3. We assign the parents of v to be the nodes of $P_{l-1}(v, p)$ in all replicas of the *SASH*. Recall that we consider vertically partitioned data as disjoint private tables where one common attribute (the ids) is public. The public edges of the *SASH* can be considered as linking id (and we continue to make no distinction between the nodes of the *SASH*, the objects, and the ids, except that the ids are keys for each party to the private attributes it holds). Each element v at level l now has up to p distinct parents associated with database elements in its vicinity.
4. Now, we create the child edges for the nodes of level $l - 1$, as follows:
   (a) For each node u of level $l - 1$, each party determines the list of distinct nodes $C(u)$ of level l that have chosen u as a parent.
   (b) Using or privacy-preserving comparison of $dist$ values, and the PP-k-NN with k set to c and the set of ids being $C(u)$, each party securely obtains the list to hold the c elements closest to u.
   (c) Now, each party connects these c nodes in $C(u)$ as the children of u.
5. Using the edges between ids, each party determines (for each node v of level l), whether it was accepted as a child of any node at level $l - 1$. If a node was accepted, then the closest node that accepted it as a child becomes the guarantor $g(v)$ of v, and v becomes a dependant of $g(v)$. This guarantor is found by invoking our PP-k-NN with $k = 1$. If the node v was not accepted as a child, we label v as an orphan node.
6. For each orphan node v at level l, a node at level $l - 1$ is needed to act as its guarantor. The node should be as close as possible to v (in terms of the distance measured), and must be unencumbered; that is, it must have fewer that the maximum allowed number of children, c. Find a guarantor for v by successively doubling the size of the candidate parents set as follows:
   (a) Set $i = 1$
   (b) Compute $P_{l-1}(v, 2^i p)$ securely as in Step 2.
   (c) If $P_{l-1}(v, 2^i p)$ has no unencumbered node, let i++ and go to Step 6b.
   (d) Otherwise, choose as the guarantor $g(v)$ the unencumbered node of $P_{l-1}(v, 2^i p)$ that is closest to v (again, using PP-k-NN, with $k = 1$ on the set of unencumbered nodes). The parties add v as a child and dependant of $g(v)$, and replace the parent of v furthest from v by $g(v)$.

The previous discussion (regarding SMC metrics and PP-k-NN as building blocks, and the algorithms of the *SASH* construction) confirm that we can produce a privacy-preserving *SASH*.

4.2 Private approximate k-NN queries

A simple and effective way to retrieve an approximation to the k-nearest neighbours of a query object q is to generate the candidate parents as in the *SASH* construction. This allows us to compute $P_1(q,k) \cup P_2(q,k) \cdots \cup P_h(q,k)$ and then we select k elements closest to q as the result of the query. Since all nodes are reachable from the root, there is a level j with more than k elements where $|P_j(q,k)| = k$. Thus, exactly k elements will be returned, (provided that the number of elements in the database is at least k).

The authors of the *SASH* propose a search pattern that improves both accuracy and search time [31] (a variable number $k_i = \max\{k^{1-\frac{h-i}{\log_2 n}}, \frac{1}{2}pc\}$ of objects is drawn from each level $1 \leq i \leq h$). The number of objects k_i selected from level i does not depend on the query object q. We can implement this privacy-preserving variant as follows.

Algorithm Privacy_Preserving_FindNearNeighbors(q,k):

1. Construct securely a set of up to $k_i > 0$ near neighbours $P_i(q,k_i)$ from among the nodes of *SASH* level i, for $1 \leq i \leq h$, as follows:
   (a) If $i = 1$, then $P_i(q,k_i)$ consists of a single node, the root (this is public knowledge).
   (b) Otherwise, $i > 1$.
      i Let $P'_i(q)$ be the set of distinct children of the nodes of $P_{i-1}(q,k_i)$ (these are publicly known edges of the *SASH*).
      ii Set $P_i(q,k_i)$ to be the k_i nodes of $P'_i(q)$ closest to q using our PP-k-NN with $k = k_i$ on the set $P'_i(q)$. If $|P'_i(q)| < k_i$, then set $P_i(q,k_i) = P'_i(q)$.
2. Using PP-k-NN again, return the k elements of $P_1(q,k_1) \cup P_2(q,k_2) \cdots \cup P_h(q,k_h)$ closest to q. If the set contains fewer than k elements, return the entire set.

4.3 Private range queries

The *SASH* can also be used to perform approximate range queries by iteratively computing approximate k-NN queries for some increasing sequence of value $k = s_1, s_2, s_3, \ldots$. For example, the size of the query could be doubled at each iteration ($s_{i+1} = 2s_i$ *for* $i > 1$). The iteration would continue until *either* an element outside the desired range is discovered (at which time all generated elements that lie within the range are reported as the solution to the range query), *or* the entire database has been visited (which occurs only when most or all of the database elements lie within the query range). If we use this doubling strategy, we can guarantee a competitive ration of two; namely, the final value k is guaranteed to be at most twice the true number of elements lying in the desired range. Because range queries are based upon privacy-preserving approximate k-NN queries, (ie, we use approximate k-NN queries for some increasing sequence of value $k = s_1, s_2, s_3, \ldots$), this immediately means that constructing a *SASH* and performing approximate k-NN queries in the privacy-preserving context suffices to have range queries in the privacy-preserving context.

While we have not shown Delete or other ADT-Dictionary operations here, the description on insertion/construction should suffice to perform the necessary extensions.

5 Performance evaluation

Algorithms which do not ensure some level of privacy will not be considered when privacy is needed. However researchers compare secure and non-secure versions to identify the overhead of privacy-preserving algorithms over a distributed non-private setting. 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.

There are solutions for a Yao-comparison without shares with complexity linear on the number of bits (Sect. 2.1). This is very efficient, and if one is prepared to disclose the outcome of comparisons in the Chessboard distance calculation (Sect. 3.2), even this metric can be computed in time linear in the number of parties and linear in the number of bits used for the metric values with a communication cost also linearly proportional to number of parties and to the number of bits. The constants involved in the O-notation are also small. Our Yao-comparison with shares trades un-feasibility of theoretical circuit evaluation for a third party who can be played by one of the other parties in settings with three or more parties. Our solution is also linear in the number of bits and the local computation time is at most a few (constant number of) operations (additions and multiplications). Using our Yao-comparison with shares and our Protocol 5 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 no more than 100, and usually about 10. This is very affordable for the additional privacy.

In the *Minkowski* metrics, one can easily see that the main cost we have is the communication cost, which is clearly affordable. In fact any other local cost (computing local projections of the metric) would also be performed in a non-private setting. The communication cost is also comparable to a non-private setting where the parties would have to communicate with each other or to a central party their local metric values. The local cost for generation of a pseudo-random number and the subtraction operation is totally subsumed with the cost of the local metric. Moreover, there are usually fewer parties than dimensions; thus, passing a value among the m parties to compute a global metric is usually well within the order of cost of computing the global value without privacy. For the combinations metrics introduced here (Sect. 3.3) (like sum/max of local metrics) the performance will be the same as for the *Minkowski* metrics for sums and Chessboard for maximums. Finally, our private cosine metric calculation is also within minimal overhead over a non-private setting. The communication cost is again essentially the same as for parties computing this metric without privacy and is still linear in the number of parties and bits of the floating-point values. There are constant (less than 8) numerical computations (addition/multiplication) between the two parties left with computing the metric value with shares. They engage in privacy-preserving computation of dot products of dimension 2.

For 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. That is, as long as computing distances are proportional to the number of dimensions, the cost (associated with k-NN queries or associative queries) is essentially the number of evaluations of distances. To perform k-NN queries, the only possible competitor to our privacy-preserving *SASH* method is the privacy-preserving version [55] of Fagin's A0 algorithm [23]. 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 [23,55] 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. Our intuition is that Fagin's algorithm must perform poorly in general because in order to perform well it requires that the cylinders around the query vector $\vec{q}$, that constitute the projection to the k-nearest neighbour in each party, contain together as few elements as k. This seems unlikely. To confirm this we evaluated the size S of the union of Fagin's AO algorithm in five well-known large data sets. The CoIL 2000 Challenge [58] dataset (Database 1) contains information on customers of an insurance company. The data consists of 86 variables and includes product usage data and socio-demographic data derived from zip area codes. We repeated the following experiment 100 times. We partitioned 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-nearest neighbours using Fagin's A0 algorithm. Table 4 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 in the table are close to or above 3, 000 while several are above 5, 000. It is rather 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. We also recorded the best and worst observed size S of the union. Rather than showing another table we present this data for $m = 8$ parties in Fig 6a. 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% of the size of the file. Similar results occur for the Census-Income Database holding multivariate PUMS census data (Database 2) from the Los Angeles and Long Beach areas for the years 1970, 1980, and 1990 (from KDD UCI repository). Combining test and training databases we get 299,285 records with 40 dimensions/attributes.

The inefficiency of A0 is also reflected in three large datasets previously used for approximate nearest neighbour queries [25].

The dataset named *Histogram* (Database 3) corresponds to a colour histogram, while the one named *Stock* (Database 4) corresponds to a stock market price. *Stock* has dimension 360

Table 4 Evaluation of the A0 algorithm on Database 1 (The Insurance Company Benchmark CoIL 2000)

k	Average size S of the union for $A0$ algorithm (95% confidence intervals)			
	Number m of parties			
	4	6	8	10
2	2,469±259	2,454 ± 197	3,503 ± 213	4,377 ± 178
3	2,589 ± 243	2,559 ± 194	3,729 ± 173	4,595 ± 142
4	2,548 ± 249	2,894 ± 192	3,903 ± 172	4,645 ± 166
5	2,753 ± 254	3,156 ± 168	4,007 ± 157	4,806 ± 145
8	2,891 ± 228	3,516 ± 151	4,334 ± 143	5,052 ± 137
10	2,982 ± 218	3,694 ± 138	4,535 ± 136	5,092 ± 117
15	3,173 ± 191	3,973 ± 121	4,708 ± 121	5,314 ± 97
20	3,100 ± 203	4,037 ± 141	4,835 ± 106	5,438 ± 55
25	3,429 ± 161	4,270 ± 127	5,004 ± 98	5,448 ± 69
50	3,704 ± 150	4,689 ± 125	5,294 ± 68	5,628 ± 44

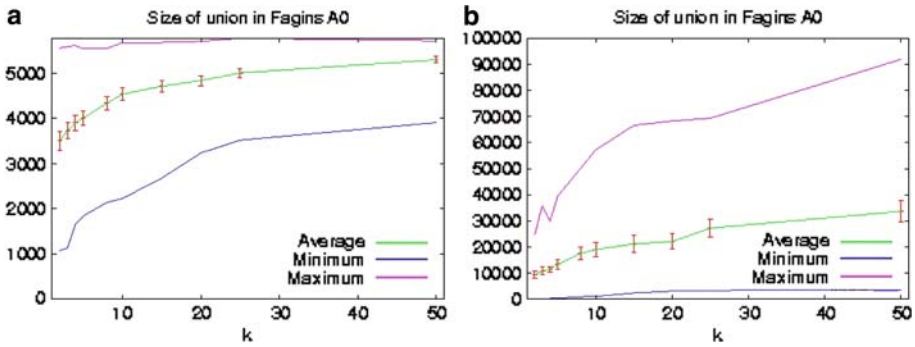

Fig. 6 Maximum, average and minimum size of S with $m = 8$ parties (average is shown with 95% confidence intervals)

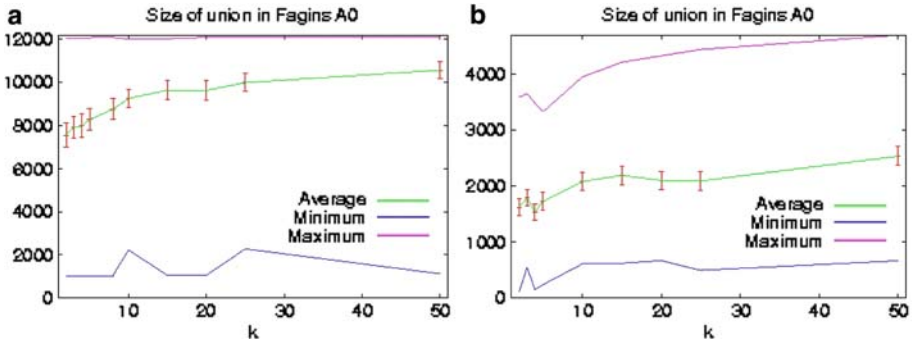

Fig. 7 Maximum, average and minimum size of S with $m = 8$ parties (average is shown with 95% confidence intervals)

and 6,500 records corresponding to different companies. For $m = 8$ parties, the observed average, minimum and maximum size of the union for AO are shown in Fig 7b for the *Stock* dataset while the results for *Histogram* dataset are shown in Fig. 7a. The histogram data set has dimension 64 and 12,103 records. The results for *histogram* show not only an average case of $O(n)$ but the worst case is essentially N. Finally, an aerial image dataset with dimension 60 and 275,465 (Database 5) records was tested and results for $m = 8$ on the size of the union for AO appear in Fig. 8. Another reason for performing this analysis is that the privacy-preserving version of Fagin's AO algorithm [55] 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 $pcN \log 2N$ [31,32] (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

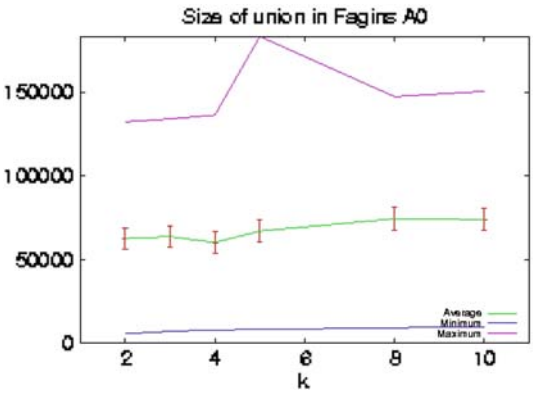

Fig. 8 Maximum, average and minimum size of S with $m = 8$ parties for Database 5 (average is shown with 95% confidence intervals)

Fagin's A0 algorithm, and will provide logarithmic response for k-NN queries and associative queries.

Naturally, the *SASH* invokes k-NN queries on small data sets. That is, the privacy-preserving *SASH* invokes Protocol 8 for small sets of size n. The natural question is why not use the searching for top k-queries in a field which requires $(n + 1) \log |F|$ rounds, as opposed to our Protocol 8 which has complexity $O(n)$.

First, clearly all steps are equivalent until these protocols receive two vectors of dimensions n known to two distinct parties. Our protocol does require $\Theta(n \log n)$ time on Alice's side for sorting and $O(n)$ time on Bob's side to add a random value to all the shares it holds and to generate the random permutation π, but the constants involved are small and clearly the process is practical. However, the search in a field requires $(n+1) \log |F|$ rounds. Typical values are $|F| = 10^6$, which makes a larger constant in the $O(n)$ for this alternative.

More importantly, each of these rounds involves a Yao-comparison with shares and a final round where each party totals n values. Clearly, the local computation is far more in this aspect alone than 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.

To further illustrate the practicality of our approach, we have implemented our Protocol 8. Overall, Protocol 8'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.

In Fig. 9a–c, we illustrate these dependencies. The implementation confirms the logarithmic performance time for m and n. The dependency from k oscillates (in a small region) but it remains bounded by constant. This is clear, because the sorting component is $O(n \log n)$ while the selection of k values is $O(k)$, with k much less than n. For communication cost, the $mn + k$ complexity is dominated by n again.

We have also implemented the *SASH* method and evaluated the performance on the same five databases used with Fagin's A0 algorithm. 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 [31, page 8]). Our results for all five databases are shown in Table 5 (with 95% confidence intervals). The *SASH* candidate generation does not depend on m, and shows remarkably small numbers for all data sets.

We have used the default parameters for the *SASH* recommended by the authors.[23] That is, we set the maximum number p of parents per node is 4 and the maximum number c

[23] We thank Dr. M. Houle for providing two initial implementations of the *SASH* from which we were able to create our experiments.

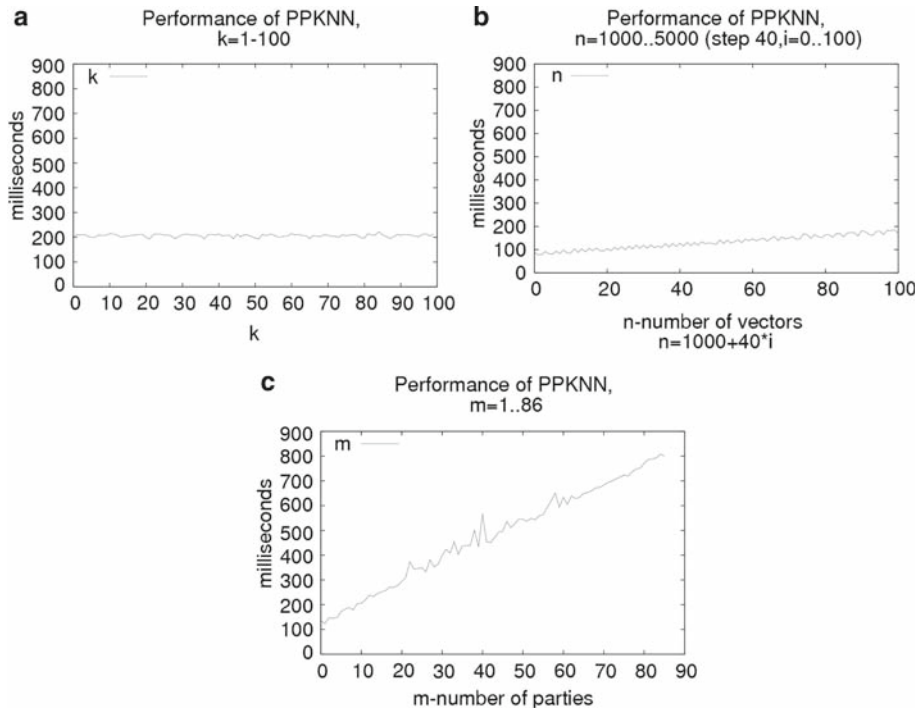

Fig. 9 Dependency on k, n and m tested on Database 1—the CoIL 2000 challenge dataset

Table 5 Number of distinct candidates generated during k-NN queries using the *SASH* (95% confidence intervals)

k	Size of the union for the *SASH* method for all five databases				
	Database 1	Database 2	Database 3	Database 4	Database 5
2	930 ± 17	1,325 ± 41	1054 ± 28	712 ± 10	1702 ± 63
3	930 ± 16	1,325 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
4	930 ± 17	1,326 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
5	930 ± 16	1,326 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
8	930 ± 16	1,325 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
10	930 ± 16	1,325 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
15	930 ± 16	1,325 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
20	930 ± 17	1,325 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
25	930 ± 16	1,326 ± 41	1,054 ± 28	712 ± 10	1,702 ± 63
50	954 ± 17	1,350 ± 42	1,074 ± 28	727 ± 11	1,737 ± 64
75	1,051 ± 20	1,461 ± 48	1,179 ± 32	789 ± 11	1,900 ± 73
100	1,157 ± 23	1,589 ± 53	1,295 ± 36	865 ± 12	2,110 ± 84

of children per node is $4p$ (that is 16) [31]. The geometric search pattern [31] is the one used for k-NN queries. The impact of this geometric pattern in our experiments is noticed in the Figs. 10a, b, 11a, b and 12a. In particular, across these figures, the number of generated

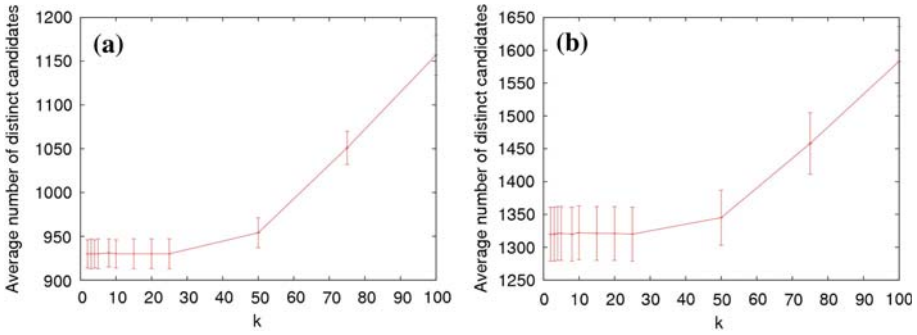

Fig. 10 Distinct candidates generation while performing k-NN queries for Database 1 and Database 2 (*bars* indicate 95% confidence intervals)

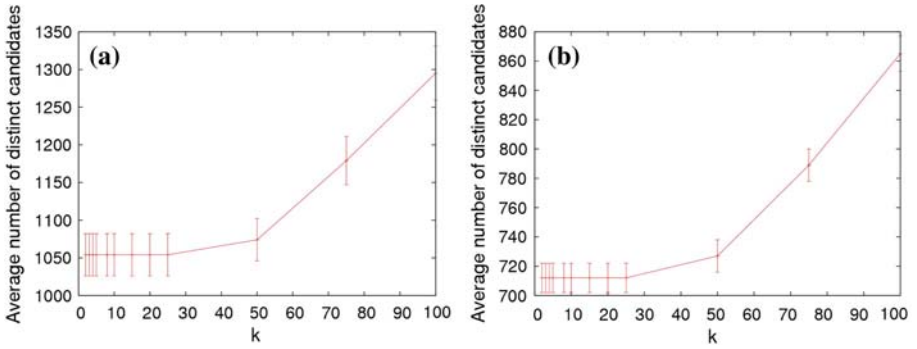

Fig. 11 Distinct candidates generation while performing k-NN queries for Database 3 and Database 4 (*bars* indicate 95% confidence intervals)

candidates remains essentially constant for k below $k = 50$. This is clear, because during the construction of the *SASH* the parameters p and c determine the size of the local collection of information about closest near neighbours for each node (see Sects. 4.1, 4.2). In particular, this influences the values of the geometric pattern used for k-NN query. In this geometric pattern, a variable number $k_i = \max\{k^{1-\frac{h-i}{\log_2 n}}, \frac{1}{2}pc\}$ of objects is drawn from each level $1 \leq i \leq h$). Let us examine, for instance, Fig. 11a. In this dataset, the value $n = 12,103$ and during the construction of the *SASH*, a total of 11 levels are contracted, so $h = 11$. When $k \leq \frac{1}{2}pc = 32$, then every $k_i = 32$ for $1 \leq i \leq h$. If $32 \leq k \leq 42$, then in the last level ($i = h$) only the number of generated candidates will be different, but this will not affect the outcome as much, due to the very small difference. Moreover, the generated extra candidates could not be distinct from the candidates already included in the list. This is exactly what happens here. Starting from $k > 42$ the last two levels produce more distinct candidates, so an increase in the overall number of generated candidates is noticeable (see Fig. 11a). An increase of the value of k will affect more levels of the *SASH*. In particular, for every node from a level above the affected levels, the number of distinct children sought will be more than $\frac{1}{2}pc = 32$. This, obviously, affects the overall result.

We alert the reader that the scale of the y-axis in Figs. 10a, b, 11a, b and 12a is logarithmic. Figure 12b collates all the data for the *SASH* with logarithmic performance.

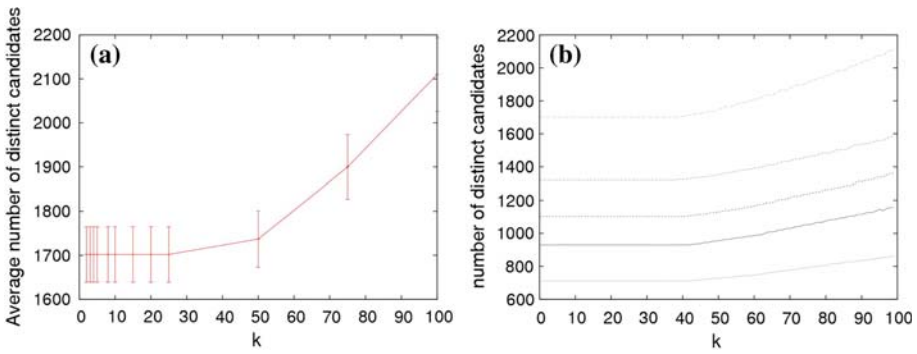

Fig. 12 Distinct candidates generation while performing k-NN queries for Database 5 and all five databases

Thus, the number of *SASH* candidates is much better than the number of Fagin's A0 candidates (at least several orders of magnitude). This demonstrates the efficiency of the *SASH* approach.

6 Final remarks

If we accept the theoretically secure Yao-comparisons protocol with shares, this paper provides formally secure solutions to computing all the metrics discussed including all *Minkowski* metrics. Moreover, our Protocol 7 is a formally secure protocol for k-NN in the semi-honest model (Theorem 7) when no parties collude. In particular, Protocol 7 is totally secure for two parties. This is a first contribution that is fulfilling in the theoretical sense. However, the protocols for metrics and Protocol 7 would be impractical by the many limitations of the theoretical solutions about Yao-comparisons with shares and because Protocol 7 requires quadratic complexity on the size of the database. We have described a Yao-comparison protocol that is secure in the formal sense (Theorem 1); however, because it uses real numbers, for its implementation we must trade-off security for efficiency. We have identified all the security risks and discussed how they can be minimized while maintaining very useful properties, like constant number of messages, and linearity on the number of bits involved (which makes it much faster than any other implementation we are aware of). As a result, we obtain an implementable version that is parameterized to achieve statistical secrecy.

While the relevance of k-NN queries have been recognized for many data-mining tasks, it has not been accepted that the current solutions for privacy-preserving computation of these queries are impractical. We have shown that the current alternative, the privacy-preserving adaptation of Fagin's AO algorithm is essentially unfeasible because of its reliance on theoretical generic solutions for Yao-comparison with shares and the assumption that the size of the union generated by Fagin's is affordable. We have shown that this option is not viable and as an alternative we provided the *SASH* for logarithmic performance on the size of the database (as opposed to linear). We believe this generic privacy-preserving version of the *SASH* is the best trade-off between privacy and computational requirements. Along the way, we converted secure algorithms for the computation of metrics into practical solutions to be used in k-NN queries (we swap into them our efficient implementation of Yao-comparisons with shares and remove the theoretical Yao-comparison). Solving k-NN queries is central for many data mining tasks. Doing so separately, rather than the union of the databases risk accuracy of results. For example, it has been shown before [21,53], for vertically partitioned

data, that the results of each party clustering separately can be radically different, and in fact incorrect, if each party clusters in their own projection. Since we have made the only element needed a $dist$ function, the applications of our approach go beyond objects codified with an id and an attribute-vector per party to unstructured data, like video and audio.

Why is the *SASH* less accepted in the context of data mining? One issue that remains to be elegantly solved for the *SASH* are altering insertions and deletions (that is a graceful dynamic behavior). Accumulating insertions and marking deleted items as such while periodically rebuilding the *SASH* may be considered sufficient for dynamic behavior. However, for privacy-preserving this is not satisfactory as some queries would be repeated on the same elements and the risk of information leak increases then.

Acknowledgments We would like to thank Joel Fenwick for valuable discussions and contributions to clarify the notation.

References

1. Amirbekyan A, Estivill-Castro V (2006) Privacy preserving DBSCAN for vertically partitioned data. In: IEEE international conference on intelligence and security informatics, ISI 2006. Lecture notes in computer science, vol 3975. Springer, San Diego, pp 141–153
2. Amirbekyan A, Estivill-Castro V (2007) The privacy of k-NN retrieval for horizontal partitioned data—new methods and applications. In: Bailey J, Fekete A (eds) Eighteenth Australasian database conference (ADC2007). Conferences in research and practice in information technology (CRPIT), CORE, Australasian Computer Society, Ballarat, Victoria, Australia, pp 33–42
3. Ankerst M, Kastenmueller G, Kriegel H, Seidl T (1999) Nearest neighbor classification in 3D protein databases. In: Proceedings of the seventh international conference on intelligent systems for molecular biology (ISMB-99), pp 34–43
4. Ankerst M, Kriegel H, Seidl T (1998) A multi-step approach for shape similarity in image databases. IEEE Trans Data Eng 10(6):996–1004
5. Arya S, Mount DM, Netanyahu NS, Silverman R, Wu AY (1994) An optimal algorithm for approximate nearest neighbor searching. In: 5th annual ACM-SIAM symposium on discrete algorithms, pp 573–582
6. Avidan S, Butman M (2006) Blind vision. In: Leonardis S, Bischof H, Pinz A (eds) Computer vision—ECCV 2006, 9th European conference on computer vision, Part III'. Lecture notes in computer science, vol 3953. Springer, Graz, pp 1–13
7. Beimel A, Ishai Y (2005) On the power of nonlinear secret-sharing. SIAM J Discrete Math 19(1):258–280
8. Bentley J (1975) Multidimensional binary search trees used for associative retrieval. Commun ACM 18(9):509–517
9. Berchtold S, Keim DA, Kriegel H-P (1996) The X-tree: an index structure for higher dimensional data. In: Proceedings of twentieth VLDB conference, pp 28–39
10. Beyer K, Goldstein J, Ramakrishnan R, Shaft U (1999) When is nearest neighbor meaningful. In: International conference on database theory, Jerusalem, Israel, pp 217–225
11. Blundo C, Masucci B, Stinson D, Wei R (2002) Constructions and bounds for unconditionally secure non-interactive commitment schemes. Designs Codes Cryptography 26:97–110
12. Breunig M, Kriegel H-P, Ng R, Sander J (2000) Lof: Identifying density-based local outliers. In: Chen W, Naughton JF, Bernstein P (eds) Proceedings of the 2000 ACM SIGMOD international conference on management of data. ACM Press, Dallas, pp 93–104
13. Cachin C (1999) Efficient private bidding and auctions with an oblivious third party. In: Proceedings of the sixth ACM conference on computer and communications security, SIGSAC, ACM Press, Singapore, pp 120–127
14. Cheng X, Dolin R, Neary M, Prabhakar S, Ravikanth K, Wu D, Agrawal D, El Abbadi E, Freeston M, Singh A, Smith T, Su J (1997) Scalable access within the context of digital libraries. In: Proceedings of the international conference on advances in digital libraries. ADL, Washington, DC, pp 70–81
15. Ciaccia P, Patella M (2000) PAC nearest neighbor queries: Approximate and controlled search in high-dimensional and metric spaces. In: Proceedings of the international conference on data engineering, San Jose, California, pp 244–255
16. Damgard I, Pedersen T, Pfitzmann B (1998) Statistical secrecy and multibit commitments. IEEE Trans Inf Theory 44(3):1143–1151

17. Du W, Atallah M (2001) Privacy-preserving cooperative statistical analysis. In: Proceedings of the seventeenth annual computer security applications conference (ACSAC), ACM SIGSAC. IEEE Computer Society, New Orleans, pp 102–110
18. Du W, Han Y-S, Chen S (2004) Privacy-preserving multivariate statistical analysis: linear regression and classification. In: BM W, Dayal U, Kamath C, Skillicorn D (eds) 2004 SIAM international conference on data mining, Lake Buena Vista, Florida, pp 222–233
19. Du W, Zhan Z (2002) Building decision tree classifier on private data. In: Estivill-Castro V, Clifton C (eds) Privacy, security and data mining, IEEE ICDM workshop proceedings, Conferences in research and practice in information technology series, Vol 14. Australian Computer Society, Sydney, pp 1–8
20. Ester M, Kriegel H, Sander S, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Simoudis E, Han J, Fayyad U (eds) Proceedings of the second international conference on knowledge discovery and data mining (KDD-96), AAAI. AAAI Press, Menlo Park, pp 226–231
21. Estivill-Castro V (2004) Private representative-based clustering for vertically partitioned data. In: Baeza-Yates R, Marroquin J, Chávez E (eds) Fifth Mexican international conference on computer science (ENC 04), SMCC. IEEE Computer Society Press, Colima, pp 160–167
22. Estivill-Castro V, Yang J (2002) Clustering Web visitors by fast, robust and convergent algorithms. Int J Found Comput Sci 13(4):497–520
23. Fagin R (1996) Combining fuzzy information from multiple systems. In: Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on principles of database systems. ACM Press, Montreal, pp 216–226
24. Faloutsos C, Ranganathan M, Manolopoulos Y (May 1994) Fast subsequence matching in time-series databases. In: Proceedings of the ACM SIGMOD international conference on management of data, Minneapolis, pp 419–429
25. Ferhatosmanoglu H, Tuncel E, Agrawal D, El Abbadi A (2002) Approximate nearest neighbor searching in multimedia databases. In: Seventh IEEE international conference on data engineering (ICDE), Germany, pp 503–511
26. Goethals B, Laur S, Lipmaa H, Mielikäinen (2005) On private scalar product computation for privacy-preserving data mining. In: Park C, Chee S (eds) Information security and cryptology—ICISC 2004. Lecture notes in computer science, vol 3506. Springer, Berlin, pp 104–120
27. Goldreich O (2004) The foundations of cryptography, vol 2, chapt General cryptographic protocols. Cambridge University Press, London
28. Goldreich O, Micali S, Wigderson A (1987) How to play any mental game (extended abstract). In: Aho A (ed) Proceedings of the nineteenth ACM annual symposium on theory of computing. ACM Press, New York, pp 218–229
29. Guttmann A (1984) R-trees: a dynamic index structure for spatial searching. In: Proceedings of ACM SIGMOD international conference on management of data, pp 47–57
30. Houle M (2003a) Navigating massive data sets via local clustering. In: Getoor L, Senator T, Domingos P, Faloutsos C (eds) Ninth ACM SIGKDD conference on knowledge discovery and data mining. ACM Press, Washington, DC, pp 547–552
31. Houle M (2003b) SASH: a spatial approximation sample hierarchy for similarity, Technical Report RT-0517, IBM Tokyo Research Laboratory, p 16
32. Houle M, Sakuma J (2005) Fast approximate similarity search in extremely high-dimensional data sets. In: Twenty first international conference on data engineering ICDE. IEEE Computer Society, Tokyo, pp 619–630
33. Ioannidis I, Grama A (2003) An efficient protocol for Yao's millionaires' problem. In: Thirty sixth Hawaii international conference on system sciences HICSS. IEEE Computer Society, Big Island, p 205
34. Kahveci T, Singh AK (2001) An efficient index structure for string databases. In: VLDB conference, Rome, Italy, pp 351–360
35. Kantarcioğlu M, Clifton C (2004) Privately computing a distributed k-nn classifier. In: Boulicaut J-F (ed) Eighth European conference on principles and practice of knowledge discovery in databases PKDD. Lecture notes in computer science, vol 3202. Springer, Berlin, pp 279–290
36. Kargupta H, Datta S, Wang Q, Sivakumar K (2005) Random-data perturbation techniques and privacy-preserving data mining. Knowl Inf Syst 7(4):387–414
37. Katayama N, Satoh S (1997) The SR-tree: an index structure for high-dimensional nearest neighbour queries. In: ACM SIGMOD conference on management of data, Tucson, USA, pp 369–380
38. Korn F, Sidropoulos N, Faloutsos C, Siegel E, Proropapas Z (1996) Fast nearest neighbor search in medical image databases. In: VLDB, Mumbai, India, pp 215–226
39. Lin X, Clifton C, Zhu M (2005) Privacy-preserving clustering with distributed EM mixture modeling. Knowl Inf Syst 8(1):68–81

40. Malkhi D, Nisan N, Pinkas B, Sella Y (2004) Fairplay—a secure two-party computation system. In: Proceedings of the thirteenth conference on USENIX security symposium, vol 13, USENIX Association, p 20
41. Mena J (2003) Investigative data mining for security and criminal detection. Butterworth-Heinemann, London
42. Morimoto Y, Aono M, Houle M, McCurley K (2003) Extracting spatial knowledge from the Web. In: International symposium on applications and the internet (SAINT), Orlando, USA, pp 326–333
43. Nielsen J, Schwatzbach M (2007) A domain-specific programming language for secure multiparty computation. In: Conference on programming language design and implementation—Proceedings on the 2007 workshop on programming languages and analysis for security, SIGPLAN, ACM Press, New York, pp 21–30
44. Peng K, Boyd C, Dawson E, Lee B (2005) An efficient and verifiable solution to the millionaire problem. In: Park C, Chee S (eds) Information security and cryptology—ICISC 2004'. Lecture notes in computer science, vol 3506. Springer, Berlin, pp 51–66
45. Rivest R (1999) Unconditionally secure commitment and oblivious transfer schemes using concealing channels and a trusted initializer (manuscript)
46. Roussopoulos N, Kelly S, Vincent F (1995) Nearest neighbor queries. In: Proceedings of the ACM SIGMOD international conference on management of data. ACM Press, San Jose, pp 71–79
47. Shaneck M, Kim Y, Kumar V (2006) Privacy preserving nearest neighbor search. In: ICDMW '06: Proceedings of the sixth IEEE international conference on data mining—Workshops. IEEE Computer Society, Washington, DC, pp 541–545
48. Shannon C (1949) Communication theory of secret systems. Bell Systems Technical Journal 28:656–715
49. Stinson DR (1995) Cryptography: theory and practice. CRC, Boca Raton
50. Subrahmanian V (1999) Principles of multimedia database systems. Morgan Kaufmann, San Francisco
51. Sung S, Liu Y, Xiong H, Ng P (2006) Privacy preservation for data cubes. Knowl Inf Syst 9(1):38–61
52. Vaidya J, Clifton CC (2002) Privacy preserving association rule mining in vertically partitioned data. In: The eighth ACM SIGKDD international conference on knowledge discovery and data mining, SIGKDD. ACM Press, Edmonton, pp 639–644
53. Vaidya J, Clifton CC (2003) Privacy-preserving *k*-means clustering over vertically partitioned data. In: Proceedings of the SIGKDD-ACM conference of data mining. ACM Press, Washington, DC, pp 206–215
54. Vaidya J, Clifton C (2004) Privacy-preserving outlier detection. In: Fourth IEEE international conference on data mining (ICDM 2004). IEEE Computer Society, Brighton, pp 233–240
55. Vaidya J, Clifton C (2005) Privacy-preserving top-*k* queries. In: Twenty first international conference on data engineering, ICDE 2005. IEEE Computer Society, Tokyo, pp 545–546
56. Vaidya J, Clifton C, Zhu M (2006) Privacy preserving data mining. Advances in information security, vol 19. Springer, New York
57. Vaidya J, Yu H, Jiang X (2008) Privacy-preserving SVM classification. Knowl Inf Syst 14(2):161–178
58. van der Putten P, van Someren M (2000) CoIL challenge 2000: the insurance company case. Technical Report 2000-09, Leiden Institute of Advanced Computer Science, Amsterdam
59. Wang K, Fung B, Yu P (2007) Handicapping attacker's confidence: an alternative to *k*-anonymization. Knowl Inf Syst 11(3):345–368
60. Witten I, Frank E (2000) Data mining—practical machine learning tools and technologies with JAVA implementations. Morgan Kaufmann, San Mateo
61. Wu X, Kumar V, Quinlan J, Ghosh J, Yang Q, Motoda H, McLachlan G, Ng A, Liu B, Yu P, Zhou Z-H, Steinbach M, Hand DJ, Steinberg D (2008) Top 10 algorithms in data mining. Knowl Inf Syst 14(1):1–37
62. Xu S, Zhang J, Han D, Wang J (2006) Singular value decomposition based data distortion strategy for privacy protection. Knowl Inf Syst 10(3):383–397
63. Yao A (1982) Protocols for secure computation. In: IEEE symposium of foundations of computer science. IEEE Computer Society, pp 160–164
64. Yu B, Ooi C, Tan K, Jagadish H (2001) Indexing the distance: an efficient method to KNN processing. In: Twenty seventh VLDB conference, Rome, Italy, pp 421–430

Author Biographies

Artak Amirbekyan is a researcher at the Earth Systems Science Computational Centre at the University of Queensland, Australia. In 2007 he graduated Ph.D. in data mining at Griffith University, Australia. He received his Master of Science degree from University of Kaiserslautern, Germany in 2005 and his Diploma in Mathematics from Yerevan State University, Armenia in 1999. Current research interests include secure multiparty computation, privacy-preserving data mining, data mining in earth sciences as well as high performance computing.

Vladimir Estivill-Castro is currently working as a Professor in the School of Information and Communication Technology, Director of Mi-PAL and Director of the Autonomous Systems Program of the Institute for Intelligent and Integrated Systems (IIIS) at Griffith University (Australia). He is also a visiting scholar at Universitat Pompeu Fabra in Spain. His main interest are algorithmic engineering, computational complexity, intelligent data analysis, privacy preserving data mining and knowledge discovery. Prof. Estivill-Castro holds a Ph.D. from the University of Waterloo in Canada and degrees from UNAM in Mexico. He serves in the editorial board of the *Journal of Research and Practice in Information Technology*, is editor in chief of the series *Conferences in Research and Practice in Information Technology* and serves on the editorial review board of the *International Journal of Data Warehousing and Mining*. He has been involved in organization of various international conferences.