Privacy-Preserving Two-Party Skyline Queries Over Horizontally Partitioned Data

Ling Chen, Ting Yu, Rada Chirkova

Privacy-Preserving Two-Party Skyline Queries over Horizontally Partitioned Data

Ling Chen[1], Ting Yu[2], and Rada Chirkova[1]

[1] Department of Computer Science, North Carolina State University, USA
[2] Qatar Computing Research Institute, Hamad Bin Khalifa University, Qatar
[1]{lchen10,rychirko}@ncsu.edu, [2]tyu@qf.org.qa

Abstract. Skyline queries are an important type of multi-criteria analysis with diverse applications in practice (e.g., personalized services and intelligent transport systems). In this paper, we study how to answer skyline queries efficiently and in a privacy-preserving way when the data are sensitive and distributedly owned by multiple parties. We adopt the classical honest-but-curious attack model, and design a suite of efficient protocols for skyline queries over horizontally partitioned data. We analyze in detail the efficiency of each of the proposed protocols as well as their privacy guarantees.

1 Introduction

Given a set of multi-dimensional vectors, a skyline query [8] is to find all the vectors that are not dominated by any other vector. Skyline queries are widely used in multi-criteria decision making applications, for example, personalized services, intelligent transport systems, location-based services and urban planning.

In this paper, we study how to answer skyline queries over sensitive data distributedly owned by multiple parties who do not fully trust each other. Similar to many past efforts on other data analysis tasks [1, 25, 28, 29], by adopting a secure multi-party computation setting, our goal is to design efficient and privacy-preserving distributed protocols to enable multiple data owners to collaboratively compute skyline queries without revealing their private inputs. Due to space limit, we focus on horizontally partitioned data where similar information about different individuals is collected in different organizations.

Our approach for privacy-preserving skyline queries over horizontally partitioned data utilizes some existing secure protocols for basic operations, such as secure comparison protocols [10,30], secure vector dominance protocols [3,19,20], secure permutation protocols [12], secure equality-testing protocols [13, 26], and secure multi-to-sum protocols [26, 27]. To securely compute skyline queries, the secure protocol needs to address two problems: (1) how to securely determine whether a vector in one party dominates a vector in the other party? (2) how to securely apply the protocol of (1) to the set of vector pairs formed by pairing a vector V_1 in one party A with each vector in the other party B, with the goal of determining whether V_1 is dominated by any vector in party B or not. To

address these two problems, straightforward compositions of such basic building blocks would not offer viable solutions. For example, existing techniques propose secure protocols to perform vector dominance checking, but they cannot be applied directly to address the first problem. The reason is that vector dominance requires each dimensional value in a vector V_1 to be strictly better than the corresponding dimensional value in another vector V_2, while the dominance used in skyline queries (referred to as skyline dominance) allows some dimensional values but not all values in V_1 to be equal to the corresponding dimensional values in V_2. Further, to address the second problem, simply invoking the protocol for the first problem for each pair of vectors when answering skyline queries would unnecessarily reveal sensitive intermediate results, i.e., V_1 is dominated or not dominated by certain vector in the other party.

To address these challenges, we propose three secure protocols. The major novelty of our protocols lies in the following aspects. First, to address the first problem, we propose a novel protocol, enhanced vector dominance protocol (EVDP). Our insight is that for two d-dimensional vectors V_1 and V_2 that are not equal in every dimension, we can obtain the same dominance results for vector dominance and skyline dominance if we improve every value in V_1 by a fixed value. Therefore, EVDP improves every value in V_1 by a fixed value to obtain V_1', and applies a vector dominance protocol over V_1' and V_2 to securely obtain the skyline dominance results of V_1 and V_2. Second, built upon EVDP, we propose three *1-to-N skyline dominance* protocols (HP.1,HP.2, and HP.3) to address the second problem. These protocols determine whether a vector in one party is dominated by any of the N vectors in the other party. By applying these protocols on every vector in each party, we can securely compute the final skyline results without disclosing non-skyline results.

These three proposed protocols provide different levels of protection of the intermediate computation results with different communication and computation costs. HP.1 applies EVDP to determine whether a vector V_a in party A is dominated by any of the n vectors $V_{b_1}, \ldots, V_{b_n}$ in party B, and terminates whenever a dominance is found or when all vectors in party B are compared with. To prevent the disclosure of which vector in party B dominates V_a, HP.1 also employs the secure permutation protocol to permute the order of the vectors in party B. However, such protocol still discloses the intermediate results about at least how many vectors in B do not dominate V_a. To prevent such disclosure, HP.2 and HP.3 improve over HP.1 with different additional costs. HP.2 requires $O(d \cdot 2^n)$ invocations of the multi-to-sum protocol, while HP.3 requires $O(d \cdot n)$ invocations of the multi-to-sum protocol for d-dimensional vectors and $O(n)$ invocations of the secure comparison protocol. When n is relatively small, HP.2 and HP.3 have comparable costs, and HP.3 is preferred when n becomes larger.

2 Skyline Queries

Skyline queries are an important class of preference queries supporting multi-criteria decision making. A skyline query over a d-dimensional data set S returns

a subset of S containing d-dimensional vectors that are not dominated by any other vector in S. We assume the existence of a total order relationship on each dimension, and refer to the dominance relationship in skyline queries as skyline dominance:

Definition 1 (Skyline Dominance). *Given two vectors $V_1 = (a_1, \ldots, a_d)$ and $V_2 = (b_1, \ldots, b_d)$, if for $i = 1, \ldots, d$, $a_i \succeq b_i$ and $\exists j$, $a_j \succ b_j$, we say that $V_1 \succ^s V_2$, where $\succ^s$ denotes skyline dominance, $\succeq$ denotes better than or equal to, and $\succ$ denotes better than.*

Consider the following example of skyline queries. Assume that a user wants to book a hotel for a conference with the goals of *inexpensive* price and *short* distance to the conference venue. Consider four hotels A ($200, 5 miles), B ($150, 2 miles), C ($120, 3 miles), and D ($150, 1 mile). Hotel C is clearly a better candidate than A because C is cheaper and closer to the venue. Therefore, we say that C dominates A. Also, D dominates B since D is as expensive as B but is closer to the venue. However, C does not dominate B since B is closer than C to the venue. Therefore, the skyline results are $\{C, D\}$.

In a straightforward approach for skyline queries, such as Block-Nested-Loops [8], a skyline vector needs to be compared against all other vectors to ensure that no other vectors dominate it. Without loss of generality, we assume that the domain of each dimension is I (the integer domain) and the larger the better, i.e., a bigger value dominates a smaller value. Each dimension can be encoded with l bits, and its value ranges from 0 to $2^l - 1$.

2.1 Horizontally Partitioned Data

We formally define skyline queries over the vectors distributed in two parties, i.e., horizontally partitioned data (HPD):

Definition 2 (Skyline Queries over HPD). *Party A has m vectors $\mathbb{V}_a = (V_{a,1}, \ldots, V_{a,m})$ and party B has n vectors $\mathbb{V}_b = (V_{b,1}, \ldots, V_{b,n})$. Vectors in both parties have the same d skyline attributes. A skyline query over HPD returns a set of vectors $\{V_i | V_i \in \mathbb{V}_a \cup \mathbb{V}_b, \nexists V_j \in \mathbb{V}_a \cup \mathbb{V}_b,$ s.t. $V_j \succ^s V_i\}$.*

To compute skyline queries over HPD, a straightforward approach is to create a list of all the vectors from two parties and compute skyline vectors over the list of the vectors. A more efficient approach is to compute skyline vectors first in each party (referred to as local skyline vectors) and then compute global skyline vectors from the local skyline vectors, since local non-skyline vectors are guaranteed not to be global skyline vectors.

Application Scenario. A state fellowship is looking for high school student candidates that have high GPA, high SAT score and high ACT score. Since the candidates are distributed over different high schools within a state, secure skyline queries over the data across different schools are required. The scores of those students that are not selected for the fellowship should not be revealed. To securely select the candidates, high schools can perform privacy-preserving skyline queries with GPA, SAT and ACT scores as the skyline attributes.

2.2 Threat Model

We consider the semi-honest adversarial model, where both parties are honest but curious. Each party knows only its own vectors. The final skyline results are known to both parties. Given a secure protocol that computes the final skyline results, both parties strictly follow the protocol specifications but are willing to learn any information leaked during execution in order to compromise privacy.

3 Secure Multi-Party Computation Building Blocks

Secure multi-party computation (SMC) [14, 30] is a framework that allows multiple parties to perform rich data analytics over their private data without revealing any information other than the output [2,3,20,21]. We next present some basic SMC protocols that we use in our proposed protocols.

Secure Comparison Protocol. Secure comparison is an important problem in cryptography and its solution serves as a primitive operation in many SMC problems. The goal of this problem is to solve the inequality between two numbers a and b, i.e., whether $a \geq b$, without revealing the actual values of a and b. Cachin [10] proposed a scheme where a semi-trusted third party provides a means for two bidders to determine whose bid is higher in zero knowledge. This scheme assumes that two bidders do not collude with each other. The communication complexity of Cachin's scheme is linear.

Secure Permutation Protocol. A secure permutation protocol is to permute the values in a vector without revealing the values of the vector and the permutation. A typical scenario is that party A has a vector $V_A = (a_1, \ldots, a_d)$ and party B has a private permutation π and a random vector $R = (r_1, \ldots, r_d)$. A secure two-party permutation protocol enables party A to obtain $\pi(V_A + R)$ without party A learning π or R; Party B also should not learn about V_A. Existing research [12] introduces a representative secure protocol based on a homomorphic public key system. The working mechanism is as follows: firstly, party A generates a key pair for a homomorphic public key system and sends the public key to party B. Using the public key, party A then encrypts V_A and sends the result $E(V_A)$ to party B. Party B computes $E(V_A) \cdot E(R) = E(V_A + R)$, permutes $E(V_A + R)$ using π, and then sends the result $\pi(E(V_A + R))$ to party A. Party A decrypts $\pi(E(V_A + R))$ by the private key and gets $D(\pi(E(V_A + R))) = \pi(D(E(V_A + R))) = \pi(V_A + R)$.

Secure Equality-Testing Protocol. An equality-testing protocol is to determine the equality between two numbers without revealing the values of two numbers. A typical solution is to apply a commutative encryption scheme to achieve secure equality testing [13, 26].

Secure Vector Dominance Protocol. Suppose party A has a vector $V_A = (a_1, \ldots, a_d)$ and party B has a vector $V_B = (b_1, \ldots, b_d)$. One existing protocol [3, 19,20] is as follows: first, party A and party B use an input disguise method to get a randomized input: $V'_A = (a'_1, \ldots, a'_{4d})$ and $V'_B = (b'_1, \ldots, b'_{4d})$. Such a disguise makes sure that for values in V'_A and V'_B, there will be the same number of $a'_i > b'_i$

situations as that of $a'_i < b'_i$ situations. Let $Z = (1, \ldots, 1, 0, \ldots, 0)$, where the first $2d$ values are all 1 (indicating $a'_i > b'_i$) and the remaining $2d$ values are all 0. Second, party B generates a random permutation π and a random vector R. Party A computes $V''_A = \pi(V'_A + R)$ using the secure permutation protocol while party B computes $V''_B = \pi(V'_B + R)$ and $Z' = \pi(Z)$. Third, party A and party B use a secure comparison protocol, such as Yao's Millionaire protocol, to compare V''_{A_i} and V''_{B_i} where $i = 1, \ldots, 4d$. Party A holds all the comparison results, $U = (u_1, \ldots, u_{4d})$. If $V''_{A_i} > V''_{B_i}$, where $i = 1, \ldots, 4d$, $u_i = 1$, otherwise, $u_i = 0$. Fourth, party A and party B use a secure equality-testing protocol to compare U with Z'. If $U = Z'$, indicating $a'_i > b'_i$ for $i = 1, \ldots, 2d$, then V_A dominates V_B. Otherwise, V_A does not dominate V_B.

Secure Multi-to-Sum Protocol. The multi-to-sum protocol [26, 27] is to convert the multiplicative sharing of a secret s to the additive sharing of s. Assume there is a secret s over a ring R and $s = a \cdot b = x + y$. $Pair(a, b)$ is called the multiplicative sharing of s. $Pair(x, y)$ is called the additive sharing of s. Initially, party A holds a while party B holds b such that $a \cdot b = s$. After executing the multi-to-sum protocol, party A holds x and party B holds y such that $x + y = s$, with no information leaked to any of them about s or the multiplicative sharing a and b.

4 Privacy-preserving Skyline Queries

4.1 Enhanced Vector Dominance Protocol (EVDP)

Existing work proposes a secure vector dominance protocol (VDP) that securely computes the vector dominance of two d-dimensional vectors, but the definition of vector dominance is more strict than skyline dominance.

Definition 3 (Vector Dominance). *Given two vectors $V_1 = (a_1, \ldots, a_d)$ and $V_2 = (b_1, \ldots, b_d)$, if for all $i = 1, \ldots, d$, $a_i \succ b_i$, then we say that $V_1 \succ^v V_2$, where $\succ^v$ denotes vector dominance and $\succ$ denotes better than.*

Table 1 illustrates the differences between vector dominance and skyline dominance using 2-dimensional vectors. As we can see from the second column ($(2, 1) \succ (1, 1)$), vector dominance and skyline dominance return different values when (1) every value in V_a is not worse than the corresponding value in V_b and at least one value in V_a is better than the corresponding value in V_b and (2) at least one and at most $d-1$ values in V_a are the same as the corresponding values in V_b. Due to such differences, we cannot directly apply secure vector dominance protocols to answer skyline queries.

To address this problem, we propose an enhanced vector dominance protocol (EVDP), which adapts VDP to support skyline dominance. EVDP accepts as input two vectors $V_a = (a_1, \ldots, a_d)$ and $V_{b_i} = (b_{i,1}, \ldots, b_{i,d})$, and improves V_{b_i} in every dimension to obtain V'_{b_i}: $V'_{b_i} = V_{b_i} + (1, \ldots, 1) = (b_{i,1} + 1, \ldots, b_{i,d} + 1)$. By performing VDP on V'_{b_i} instead of V_{b_i}, EVDP obtains the same dominance results as skyline dominance except when V_{a_i} and V_{b_i} are exactly the same, as shown in the first column in Table 1. In other words, if we assume that V_{b_i} and

Table 1: Differences between Vector Dominance and Skyline Dominance

Input: V_a, V_b	$(1,1) \succ (1,1)$	$(2,1) \succ (1,1)$	$(2,2) \succ (1,1)$	$(2,0) \succ (1,1)$
Vector Dominance	F	F	T	F
Skyline Dominance	F	T	T	F
Enhanced Vector Dominance	T	T	T	F

V_a are not the same in every attribute (referred to as the *inequality assumption*), then the results of EVDP is the same as skyline dominance's.

Checking the inequality assumption allows both parties to know that they have a set of common vectors. We next analyze how such disclosure can be used to infer side information about the other party. First, if a vector V_1 in party A and a vector V_2 in party B are equal, there exists no vector in either party dominating V_1 or V_2, and V_1 and V_2 must belong to the final skyline results, which are known to both parties. Such disclosure cannot help a party infer the values of the remaining vectors in another party. Second, ideally, even if party A sees a vector of its local skyline results V_1 is in the final results, it should not know whether party B has the same vector. Thus, knowing the existence of V_2 can be used by A to infer that the local skyline results of party B do not contain the vectors that can be dominated by V_2. However, such disclosure cannot be directly used by A to infer the complete data distribution of B's vectors.

To securely test the inequality assumption, we adopt equality-testing protocol, which leverages homomorphic encryption to securely identify from two parities all the vectors that have the same attribute values. For these identified vectors, we can safely output them as the final skyline results. Also, since these vectors do not dominate any other vector V' in the local skyline results (otherwise V' would not appear in the local skyline results), we can safely exclude them from the skyline dominance testing against the remaining vectors.

For computing the final skyline results from the remaining vectors, we propose three secure protocols based on EVDP.

4.2 1-to-N Skyline Dominance

As discussed earlier, a straightforward protocol that applies EVDP on each pair of vectors in both parties discloses significant intermediate results. To prevent such disclosure, we define *1-to-N skyline dominance* as the primitive operation in privacy-preserving skyline queries over HPD.

Definition 4 (1-to-N Skyline Dominance). *Party A has one vector* $V_a = (a_1, \ldots, a_d)$ *and party B has n vectors* $V_{b_1} = (b_{1,1}, \ldots, b_{1,d}), \ldots, V_{b_n} = (b_{n,1}, \ldots, b_{n,d})$. *1-to-N skyline dominance returns false if* $\nexists V_{b_i}$ *s.t.* $V_{b_i} \succ^s V_a$*; otherwise, returns true.*

Based on this definition, we propose three secure horizontal 1-to-N skyline dominance protocols, referred to as HP.1, HP.2, and HP.3, which securely compute skyline dominance between a vector V_a in Party A and n vectors in party B. To

obtain the final skyline results, these protocols are applied $m+n$ times on each vector in parties A and B. One party can easily learn the number of vectors in the local skyline results in the other party (i.e., m or n) during the computation. To prevent the disclosure of m and n, both parties can generate random number of dummy vectors that are guaranteed to be pruned in the computation of final skyline results. The dummy vectors can be generated by randomly selecting out-of-domain values. For example, when each element in a vector is in the range $[0, 1000)$, the dummy vectors can be generated within the range $(-\infty, 0)$. We next present the details of the three protocols.

4.3 Secure Horizontal 1-to-N Skyline Dominance Protocol (HP.1)

To determine whether V_a is dominated by any of the n vectors from party B (1-to-N Skyline Dominance), HP.1 first pairs V_a with each of the vectors in party B and gets pairs of vectors $(V_a, V_{b_1}), \ldots, (V_a, V_{b_n})$. The steps of HP.1 are shown in Algorithm 1. To securely compute skyline queries over HPD, HP.1 prevents two types of disclosure: (1) for each pair (V_a, V_{b_i}), which values and how many values in V_{b_i} dominate the corresponding values in V_a (Steps 1-4); (2) which vector in $\{V_{b_1}, \ldots, V_{b_n}\}$ dominates V_a (Steps 5-6).

In Step 1, party B prepares each of its vectors for EVDP by increasing the values in the vectors.

Steps 2-4 include input disguise, secure permutation and secure comparison, which are adapted from VDP. In Step 2, both parties A and B disguise their vectors to prevent the disclosure of how many values in V_a are better than the values in V_{b_i}. We adopt the disguise algorithm based on [3]: given party A's vector $V_a = (a_1, \ldots, a_d)$, it generates a $4d$-dimension vector $V'_a = (a'_1, \ldots, a'_{4d})$, such that

$$a'_1 = 2a_1, \quad \ldots, \quad a'_d = 2a_d, \tag{1}$$
$$a'_{d+1} = 2a_1 + 1, \quad \ldots, \quad a'_{2d} = 2a_d + 1, \tag{2}$$
$$a'_{2d+1} = -2a_1, \quad \ldots, \quad a'_{3d} = -2a_d, \tag{3}$$
$$a'_{3d+1} = -(2a_1 + 1), \quad \ldots, \quad a'_{4d} = -(2a_d + 1). \tag{4}$$

Given party B's vector $V_{b_i} = (b_{i,1}, \ldots, b_{i,d})$, it generates a $4d$-dimension vector $V'_{b_i} = (b'_{i,1}, b'_{i,2}, \ldots, b'_{i,4d})$, such that

$$b'_{i,1} = 2b_{i,1} + 1, \quad \ldots, \quad b'_{i,d} = 2b_{i,d} + 1, \tag{5}$$
$$b'_{i,d+1} = 2b_{i,1}, \quad \ldots, \quad b'_{i,2d} = 2b_{i,d}, \tag{6}$$
$$b'_{i,2d+1} = -(2b_{i,1} + 1), \quad \ldots, \quad b'_{i,3d} = -(2b_{i,d} + 1), \tag{7}$$
$$b'_{i,3d+1} = -2b_{i,1}, \quad \ldots, \quad b'_{i,4d} = -2b_{i,d}. \tag{8}$$

This disguise scheme has the following interesting property: for a pair (V'_a, V'_{b_i}), there will be the same number of $a'_j > b'_{i,j}$ situations as that of $a'_j < b'_{i,j}$ situations when $a_j = b_{i,j}$ or $a_j > b_{i,j}$ or $a_j < b_{i,j}$. Thus, we cannot infer how many values in V_1 are greater than the values in V_2 by simply counting the

number of "1"s in the comparison vector [3]. Also, we can see that when the d-dimensional vector V'_a dominates V'_{b_i}, the comparison vector must be a $4d$-dimensional vector where the first $2d$ attribute values are all "1" and the remaining $2d$ attribute values are all "0". Thus, we construct $Z = (\underbrace{1, \ldots, 1}_{2d}, \underbrace{0, \ldots, 0}_{2d})$ in Step 2, which is used to determine whether V_a is dominated by V_{b_i} by comparing the permuted Z with the comparison vector noted as $\mathbb{U}$ in HP.1 (Step 6).

In Step 3, HP.1 uses the secure permutation protocol to prevent B from learning the order of the vectors used to perform skyline dominance checking with V_a. Party A generates n random permutations and n random vectors. These permutations and random vectors are used to permute the disguised vectors V'_a, V'_{b_i} and Z. We use n random permutations instead of one to prevent adversaries from guessing the data distribution of the skyline attribute values.

In Step 4, for the ith pair ($V''_{a,i}$,V''_{b_i}), party B applies a secure comparison protocol to obtain comparison results U_i, where $U_i = (u_{i,1}, \ldots, u_{i,4d})$, $u_{i,j} = 1$ if $b''_{i,j} > a''_{i,j}$, and $u_{i,j} = 0$ if $b''_{i,j} \leq a''_{i,j}$ for $j = 1, \ldots, 4d$. If we simply run a secure equality testing protocol for each pair of Z'_i in $\mathbb{Z}$ and U_i in $\mathbb{U}$, we will disclose the intermediate results about whether V_a is dominated by V_{b_i} or not.

To prevent disclosure of the intermediate results, HP.1 applies a secure permutation protocol on $\mathbb{Z}$ and $\mathbb{U}$ (Step 5). Party A generates a new random permutation π_{n+1} to obtains $\mathbb{Z}'$, and party B obtains $\mathbb{U}'$ by using π_{n+1}. Then $\mathbb{Z}'$ is compared with $\mathbb{U}'$ by using a secure equality testing protocol: the testing terminates if there is any $U'_i = Z''_i$ or none can be found.

Example. Assume that Party A has $V_a = (1, 1)$, and Party B has three vectors: $V_{b_1} = (3, 1)$, $V_{b_2} = (2, 2)$, and $V_{b_3} = (4, 0)$. After Step 1, we have $V_{b_1} = (4, 2)$, $V_{b_2} = (3, 3)$, and $V_{b_3} = (5, 1)$. We then perform input disguise (Step 2): Party A has $V'_a = (2, 2, 3, 3, -2, -2, -3, -3)$ and Z, and Party B has $V'_{b_2} = (7, 7, 6, 6, -7, -7, -6, -6)$. We omit the detailed transformations of V'_{b_1} and V'_{b_3} due to space limitations. In Step 3, assume that $R_2 = (1, 1, 1, 1, 1, 1, 1, 1)$ and π_2 simply switches the first $4d$ values with the last $4d$ values: Party A has $V''_{a,2} = (-1, -1, -2, -2, 3, 3, 4, 4)$ and $Z'_2 = (0, 0, 0, 0, 1, 1, 1, 1)$, and Party B has $V''_{b_2} = (-6, -6, -5, -5, 8, 8, 7, 7)$. In Step 4, by applying secure comparison on $V''_{a,2}$ and V''_{b_2}, Party B obtains $U_2 = (0, 0, 0, 0, 1, 1, 1, 1)$. In Step 5, assume that $R'_2 = (2, 2, 2, 2, 2, 2, 2, 2)$ and π_4 permutes $(1, 2, 3)$ to $(2, 1, 3)$: Party A obtains $Z''_2 = (2, 2, 2, 2, 3, 3, 3, 3)$ and $\mathbb{Z}' = (Z''_2, Z''_1, Z''_3)$, and Party B has $U'_2 = (2, 2, 2, 2, 3, 3, 3, 3)$ and $\mathbb{U}' = (U'_2, U'_1, U'_3)$ (note that Party B does not know the order of items in $\mathbb{U}'$). In Step 6, Z''_2 and U'_2 are first compared and since they are equal, we know that V_a is at least dominated by V_{b_2}.

Communication and Computation Cost. HP.1's communication cost includes the costs for applying the secure permutation protocol, the secure comparison protocol, and the secure equality testing protocol.

For a vector in party $A(B)$, HP.1 needs to apply $n + 1$ $(m + 1)$ times of the secure permutation protocol. We assume each dimension in a vector is an integer that can be encoded with l bits, and its value ranges from 0 to $2^l - 1$. The communication cost of applying one secure permutation protocol is $O(d \cdot l)$ as it

uses the homomorphic public key system. As HP.1 is applied m times for vectors in party A and n times for vectors in party B, the cost of the secure permutation protocol is $O(n \cdot m \cdot d \cdot l)$. The computation cost of secure permutation depends on the adopted protocol. A representative protocol (see Section 3) requires $O(1)$ encryption/decryption, $O(1)$ modular multiplication to obtain $E(V_A) \cdot E(R)$, and $O(1)$ application of the permutation π, whose cost is $O(d)$ for permuting values in a vector.

Algorithm 1 Secure 1-to-N Skyline Dominance Protocol (HP.1)

Input: Party A has one vector $V_a = (a_1, \ldots, a_d)$ and party B has n vectors $V_{b_1} = (b_{1,1}, \ldots, b_{1,d}), \ldots, V_{b_n} = (b_{n,1}, \ldots, b_{n,d})$.
Output: Whether V_a is dominated by any of the n vectors in party B or not.

1: EVDP preparation:
Party B : **for** $i = 1$ **to** n **do**
$V_{b_i} = \textsc{Improve}(V_{b_i})$
2: Input disguise:
Party A : $V'_a = \textsc{Disguise}(V_a)$, $Z = (\underbrace{1, \ldots, 1}_{2d}, \underbrace{0, \ldots, 0}_{2d})$
Party B : **for** $i = 1$ **to** n **do**
$V'_{b_i} = \textsc{Disguise}(V_{b_i})$
3: Secure permutation:
Party A : **for** $i = 1$ **to** n **do**
Generate π_i and R_i, $V''_{a,i} = \pi_i(V'_a + R_i)$, $Z'_i = \pi_i(Z)$
$\mathbb{Z} = (Z'_1, \ldots, Z'_n)$
Party B : **for** $i = 1$ **to** n **do**
$V''_{b_i} = \textsc{SecurePermutation}(\pi_i, V'_{b_i}, R_i)^*$
4: Secure comparison:
Party B : **for** $i = 1$ **to** n **do**
$U_i = \textsc{SecureComparison}(V''_{a,i}, V''_{b_i})$
$\mathbb{U} = (U_1, \ldots, U_n)$
5: Secure permutation:
Party A : Generate π_{n+1}
for $i = 1$ **to** n **do**
Generate R'_i, $Z''_i = Z'_i + R'_i$
$\mathbb{Z}' = \pi_{n+1}(Z''_1, \ldots, Z''_n)$, $\mathbb{R} = (R'_1, \ldots, R'_n)$
Party B : $\mathbb{U}' = \textsc{SecurePermutation}(\pi_{n+1}, \mathbb{U}, \mathbb{R})$
6: Secure equality testing:
for $i = 1$ **to** n **do**
if $\textsc{SecureEqualityTest}(U'_i, Z''_i)$ **then return true**
return false

The secure comparison protocol is applied to compare corresponding attribute values in two vectors, and thus we need to apply at most $n \cdot 4d$ times of the

* Using the secure permutation protocol, party B gets $V''_{b_i} = \pi_i(V'_{b_i} + R_i)$.

secure comparison protocol. According to Cachin [10], the communication cost of Cachin's protocol to compare input numbers is $O(l)$. By applying $n+m$ times of HP.1, the communication cost of the secure comparisons is $O((n+m)\cdot d\cdot l)$. The computation cost of one secure comparison includes the costs of one generation of the garbled circuits and one evaluation of the garbled circuits, which also includes the adopted oblivious transfer protocols and encryption/decryption of the values for the gates in the circuits [10,18,30]. Latest works on garbled circuits show that 16K-bit integer comparison can be computed within 0.5s [17, 22].

The secure equality testing protocol uses the homomorphic public key system. Thus, the communication cost of the adapted equality testing protocol for evaluating the inequality assumption and the equality testing for determining the skyline dominance are both $O((n+m)\cdot d\cdot l)$. The computation cost is $O(n+m)$ invocations of encryption and $O(n\cdot m\cdot d)$ of comparisons to identify the equal encrypted vectors.

Security Analysis. In HP.1, Steps 1-4 protect the comparison results of a pair (V_a,V_{b_i}). In Step 1, the result is computed locally in party B and no information is disclosed. The input disguise in Step 2 makes sure that if $a_j > b_{i,j}$, we will have $a'_j > b'_{i,j}$, $a'_{d+j} > b'_{i,d+j}$, $a'_{2d+j} < b'_{i,2d+j}$, and $a'_{3d+j} < b'_{i,3d+j}$, preventing the disclosure of the comparison results in Step 4. In Step 3, party A generates the permutations and the random vectors, and thus after permutation party B loses track of the order of the values in the vectors. Party A does not know the values of U_i and is not aware of the comparison results.

Steps 5-6 protect the comparison results of the pairs $(V_a,V_{b_i}), \ldots, (V_a,V_{b_n})$. In Step 5, party A generates the permutation π_{n+1} and the random vector $\mathbb{R}$, and thus after permutation party B loses track of the order of the vectors in $\mathbb{U}$. On the other hand, party A knows the order of the vectors $V''_{a,i}, (i = 1,\ldots,n)$ after the permutation. However, since all the $V''_{a,i}$ come from V_a, party A cannot infer more information from the order other than what party A originally knows.

However, step 6 could disclose the intermediate results of at least how many vectors in party B *do not* dominate V_a. In Step 6, parties A and B runs secure equality testing until a pair (V_a, V_{b_i}) is found to be equal. If such a pair is not found, then V_a is not dominated by any vector in party B and no intermediate information is disclosed. But if such a pair is found in the ith secure equality testing, then both parties know that at least i-1 vectors in party B do not dominate V_a in party A.

4.4 Enhanced Secure Horizontal 1-to-N Skyline Dominance Protocol (HP.2)

To prevent the disclosure in HP.1, i.e., V_a in party A is not dominated by at least $i-1$ vectors in party B, we propose an enhanced secure horizontal 1-to-N skyline dominance protocol (referred to as HP.2), which replaces the secure equality testing (Step 6 in Algorithm 1) with a secure protocol to compute a product of the polynomials: $\prod_{i=1}^{n}(Z''_i - U'_i)$. If there exists i such that $Z''_i = U'_i$,

the product is 0, i.e., V_a in party A is dominated by at least one vector in party B and V_a is not a vector in the final skyline results.

Algorithm 2 Enhanced Secure 1-to-N Skyline Dominance Protocol (HP.2)

Input: Party A has one vector $V_a = (a_{1,1}, \ldots, a_{1,d})$ and party B has n vectors $V_{b_1} = (b_{1,1}, \ldots, b_{1,d}), \ldots, V_{b_n} = (b_{n,1}, \ldots, b_{n,d})$.

Output: Whether V_a is dominated by any of the n vectors in party B or not.

1: Steps 1 - 5 from HP.1.

2: Secure compute $\prod_{i=1}^{n}(Z_i'' - U_i')$:

$$\prod_{i=1}^{n}(Z_i'' - U_i') = \prod_{i=1}^{n}(Z_i'') + \underbrace{\prod_{i=1}^{n-1}(Z_i'') \cdot (-U_n') + \ldots + Z_1'' \cdot \prod_{i=2}^{n}(-U_i')}_{2^n-2 \text{ multiplicative sharings: } MS_1, \ldots, MS_{2^n-2}} + \prod_{i=1}^{n}(-U_i')$$

Party A : **for** $i = 1$ **to** $2^n - 2$ **do**

$$X_{a_i} = \textsc{MultitoSum}(MS_i)$$

$$\mathbb{A}_z = \prod_{i=1}^{n}(Z_i'') + \sum_{i=1}^{2^n-2} X_{a_i}$$

Party B : **for** $i = 1$ **to** $2^n - 2$ **do**

$$X_{b_i} = \textsc{MultitoSum}(MS_i)$$

$$\mathbb{B}_z = \prod_{i=1}^{n}(-U_i') + \sum_{i=1}^{2^n-2} X_{b_i}$$

3: Equality testing:
if $\mathbb{A}_z + \mathbb{B}_z = 0$ **then return true**
else return false

Secure evaluation of the product of polynomials can be achieved by leveraging the existing secure multi-to-sum protocol. The steps of the algorithm is shown in Algorithm 2. We first perform the polynomial expansion and obtain a sum of 2^n products. Each product can be converted from the multiplicative sharing to the additive sharing by the secure multi-to-sum protocol. Each party sums all the additive sharing together. Then each party shares the sum and securely computes the final product value of $\prod_{i=1}^{n}(Z_i'' - U_i')$. If it is equal to 0, there must be at least one factor, i.e., $(Z_i'' - U_i')$, being 0, indicating that V_a in party A is dominated.

Let us walk through the process of polynomial expansion by an example where $n = 2$. Party A holds $\mathbb{Z}' = (Z_1'', Z_2'')$ while party B holds $\mathbb{U}' = (U_1', U_2')$. By polynomial expansion, $\prod_{i=1}^{2}(Z_i'' - U_i') = Z_1'' \cdot Z_2'' - Z_1'' \cdot U_2' - U_1' \cdot Z_2'' + U_1' \cdot U_2'$. Party A holds the multiplicative value of $Z_1'' \cdot Z_2''$ while party B holds the value of $U_1' \cdot U_2'$. We use the secure multi-to-sum protocol to obtain the additive sharing X_a and X_b such that $X_a + X_b = Z_1'' \cdot U_2'$, where X_a is held by party A and X_b is held by party B. We do the same for $U_1' \cdot Z_2''$ with Y_a is held by party A and Y_b

is held by party B. Therefore, party A holds the sum $\sum_A = Z_1'' \cdot Z_2'' + X_a + Y_a$ and party B holds the sum $\sum_B = U_1' \cdot U_2' + X_b + Y_b$. Then we adopt the secure sum protocol to obtain $\sum = \sum_A + \sum_B$. If $\sum = 0$, V_a is dominated.

Communication and Computation Cost. HP.2 includes the private permutation protocol, the secure comparison protocol, and the secure equality testing protocol. The analyses of these three protocols are the same as those in HP.1 except that HP.2 replaces Step 6 of HP.1 with the secure evaluation of the product $\prod_{i=1}^{n}(Z_i'' - U_i')$. After the polynomial expansion, this product of polynomials is expanded into 2^n items. Among them, there are $d \cdot (2^n - 2)$ items requiring the multi-to-sum protocol to securely convert the multiplicative sharing to the additive sharing.

To compute the final skyline results, HP.2 is applied $m + n$ times. For party A, it requires $m \cdot d \cdot (2^n - 2)$ invocations of the multi-to-sum protocol. For party B, it requires $n \cdot d \cdot (2^m - 2)$ invocations of the multi-to-sum protocol. In total, the complexity is $O(m \cdot d \cdot (2^n - 2) + n \cdot d \cdot (2^m - 2))$ invocations of the multi-to-sum protocol. Even though the cost of the multi-to-sum protocol is $O(l)$ invocations of OT_2^1 oblivious transfer of strings whose cost is $O(l)$ [26, 27], the exponential invocations of the multi-to-sum protocol are expensive.

Security Analysis. HP.2 provides strong security guarantee. Compared with HP.1, in Step 6 of HP.2, the evaluation of $\sum = \prod_{i=1}^{n}(Z_i'' - U_i')$ is protected by the secure multi-to-sum protocol. If $\sum$ is 0, both parties know that V_a is dominated by at least one vector of B. But they cannot infer which vectors in B dominate V_a. If $\sum$ is not 0, both parties know that V_a is not dominated by any vector of B, and thus V_a becomes a vector in the final skyline results.

4.5 Alternative Enhanced Secure Horizontal 1-to-N Skyline Dominance Protocol (HP.3)

As discussed above, the use of polynomial expansion in HP.2 incurs exponential communication and computation costs. To reduce the cost, we further propose a more efficient protocol HP.3. The insight of HP.3 is to replace the secure evaluation of the product $\prod_{i=1}^{n}(Z_i'' - U_i')$ in HP.2 with the secure computation of $(Z_i'' - U_i')^2$. Since $(Z_i'' - U_i')^2$ are all greater than or equal to 0, if the min value of $(Z_i'' - U_i')^2$ is 0, V_a in party A is dominated; otherwise, V_a is not dominated. We next discuss the three new steps of HP.3 in Algorithm 3.

In Step 2, each of $(Z_i'' - U_i')^2$ is expanded into a polynomial, i.e., $(Z_i'' - U_i')^2 = (Z_i'')^2 - 2 \cdot Z_i'' \cdot U_i' + (U_i')^2$, where $2 \cdot Z_i'' \cdot U_i'$ can be converted to the additive sharing using the multi-to-sum protocol and then party A holds X_{a_i} and party B holds X_{b_i}. Then each of $(Z_i'' - U_i')^2$ can be transformed into the additive sharing where party A holds $\mathbb{A}_i = (Z_i'')^2 + X_{a_i}$ and party B holds $\mathbb{B}_i = (U_i')^2 + X_{b_i}$. Therefore, party A holds $\mathbb{A} = (\mathbb{A}_1, \ldots, \mathbb{A}_n)$ and party B holds $\mathbb{B} = (\mathbb{B}_1, \ldots, \mathbb{B}_n)$.

In Step 3, we securely find the min value of $\mathbb{A}_i + \mathbb{B}_i$ using $n-1$ times of secure comparisons. For example, we test whether $\mathbb{A}_1 + \mathbb{B}_1 < \mathbb{A}_2 + \mathbb{B}_2$, i.e., $\mathbb{A}_1 - \mathbb{A}_2 < \mathbb{B}_2 - \mathbb{B}_1$. If $\mathbb{A}_1 + \mathbb{B}_1 < \mathbb{A}_2 + \mathbb{B}_2$, we keep the pair $(\mathbb{A}_1, \mathbb{B}_1)$ to compare with the next pair $(\mathbb{A}_3, \mathbb{B}_3)$. It requires $n-1$ times of secure comparisons to find the min value $\mathbb{A}_z + \mathbb{B}_z$.

In Step 4, both parties reveal the values of $\mathbb{A}_z$ and $\mathbb{B}_z$, whose sum is minimum among all, and compute $\mathbb{A}_z + \mathbb{B}_z$, and then test whether the min sum value is equal to 0 or not. If it is 0, V_a in party A is dominated. Otherwise, V_a is a global skyline vector.

Algorithm 3 Alternative Enhanced Secure 1-to-N Skyline Dominance Protocol (HP.3)

Input: Party A has one vector $V_a = (a_{1,1}, \ldots, a_{1,d})$ and party B has n vectors $V_{b_1} = (b_{1,1}, \ldots, b_{1,d}), \ldots, V_{b_n} = (b_{n,1}, \ldots, b_{n,d})$.
Output: Whether V_a is dominated by any of the n vectors in party B or not.

1: Steps 1 - 5 from HP.1.
2: $(Z_i'' - U_i')^2$ computation:
Party A : **for** $i = 1$ **to** n **do**
$X_{a_i} = \textsc{MultitoSum}(Z_i'', U_i')$
$\mathbb{A}_i = (Z_i'')^2 + X_{a_i}$
Party B : **for** $i = 1$ **to** n **do**
$X_{b_i} = \textsc{MultitoSum}(Z_i'', U_i')$
$\mathbb{B}_i = (U_i')^2 + X_{b_i}$
3: Min value identification:
Party A : $\mathbb{A}_z = \mathbb{A}_1$
Party B : $\mathbb{B}_z = \mathbb{B}_1$
for $i = 2$ **to** n **do**
if $\textsc{SecureGreaterThan}(\mathbb{A}_z - \mathbb{A}_i,\ \mathbb{B}_i - \mathbb{B}_z)$ **then**
Party A : $\mathbb{A}_z = \mathbb{A}_i$
Party B : $\mathbb{B}_z = \mathbb{B}_i$
4: Equality testing:
if $\mathbb{A}_z + \mathbb{B}_z = 0$ **then return true**
else return false

Communication and Computation Cost. The analysis of Step 1 in Algorithm 3 is the same as HP.1. For Step 2, it requires $O(d \cdot n)$ invocations of the multi-to-sum protocol to securely convert the multiplicative sharing $Z_i'' \cdot U_i'$ into the additive sharing. For Step 3, it requires $O(n)$ invocations of the secure comparisons to securely identify the pair of $(\mathbb{A}_z, \mathbb{B}_z)$. For Step 4, there is no secure protocol applied and the cost can be ignored. Thus, HP.3 reduces the cost of HP.2 from the exponential invocations of the multi-to-sum protocol to the linear invocations of the multi-to-sum protocol plus the linear invocations of the secure comparison protocol.

Security Analysis. Step 1 of HP.3 is the same as Steps 1-5 of HP.1, and thus protects the comparison results of the pairs (V_a, V_{b_i}). Steps 2-3 protect the information of how many vectors in party B may dominate V_a. The multi-to-sum

protocol in Step 2 makes sure that both parties cannot know the sum values of each pair $(\mathbb{A}_i, \mathbb{B}_i)$ while searching for the min sum $\mathbb{A}_z + \mathbb{B}_z$. Assume that after Step 3, both parties know that the ith pair produces the min sum $(\mathbb{A}_z, \mathbb{B}_z)$. Party B does not know the order of vectors as Step 5 in HP.1 (Algorithm 1) permutes the order. Hence, party B cannot infer which vector produces the value $\mathbb{B}_z$. For party A, all vectors are derived from V_a and thus no more information can be inferred by party A regarding which vector in party B contributes to the value $\mathbb{B}_z$.

In Step 4, both parties reveal the values of $(\mathbb{A}_z, \mathbb{B}_z)$ to compute the minimum sum. If $\mathbb{A}_z + \mathbb{B}_z = 0$, both parties learn that V_a is dominated. Party A can learn such result from the output as well. Party B can learn what vectors in A are global skyline vectors from the output and V_a is not among the global skyline vectors. But B does not know the values of V_a, and thus cannot know which vector in A is pruned. In addition, with dummy vectors included in the local skyline results of both parties, knowing that V_a is pruned cannot be used by each party to learn the exact number of the pruned vectors. If $\mathbb{A}_z + \mathbb{B}_z \neq 0$, both parties can learn V_a is not dominated and is a vector in the final skyline results, which can be learned from the output anyway. Since A generates the permutations, A knows the order of vectors in B and the ith vector of B is selected for $(\mathbb{A}_z, \mathbb{B}_z)$. However, A does not know the values of any vector from B and such information cannot be combined with $\mathbb{B}_z$ to infer the value of V_{b_z}. For B, since the permutation π_{n+1} is generated by A, B does not know which vector of B is selected for $(\mathbb{A}_z, \mathbb{B}_z)$, and $\mathbb{A}_z$ cannot be used by B to infer the value of V_a.

4.6 Extension to N-Parties

All the proposed protocols (HP.1,HP.2, and HP.3) can be extended to support the secure communication and computation of N parties $(N > 2)$: (1) for party P_1 from N parties, apply the protocols on P_1 and every other $N-1$ parties, and only output the results of P_1 if the results are not dominated by any vector in the other $N-1$ parties; (2) P_2 first selects only the vectors that are not dominated by the results output by P_1, then runs the protocols with every other $N-2$ parties, and outputs the results of P_2; (3) repeat (2) for the remaining parties.

5 Related Work

Yao [30] first proposed the two-party comparison problem (Yao's millionaire protocol) and developed a provably secure solution. Ben-Or et al. [5] and Chaum [11] then proposed secure protocols for computing addition and multiplication (XOR and AND). More recently, the advances in the theory of secure multi-party computations [10, 15, 16] proved that any multi-party function can be computed securely with certain amount of computation cost.

With these primitives, finding an efficient and practical solution for a specific problem is still not trivial. A line of research has focused on developing efficient

secure multi-party communication protocols for specific functions. There exists research work on more generic primitives such as set operations [6, 23], top-k queries [28]. More recent work also focuses on secure protocols for vector dominance [19,20]. These protocols focus on finding more efficient solutions to specific problems, which can provide building blocks for more complex applications, such as the problem of skyline queries that our work focuses on.

Another important line of research focuses on creating frameworks and specialized programming languages to implement and run secure multi-party computation protocols. The early approaches include FairplayMP [4], Sharemind [7], and SEPIA [9], which implement generic MPC frameworks that support a similar set of primitives, such as addition, multiplication, comparisons and equality testing. These frameworks either provide specialized programming languages to facilitate the security programming, or allow users to program using standard programming languages and library calls. With the advances of research on garbled circuits [18, 24], more efficient framework, ObliVM [22], has been built to make secure computations easier and faster. This framework provides a domain-specific language designed for compiling programs into efficient oblivious representations suitable for secure computations. It also provides MapReduce abstractions that facilitate the parallelization of the compiled program.

Unlike this line of research work that focuses on building efficient generic secure computation frameworks, our work focuses on designing efficient secure protocols that minimize the costs of communication and computation while preserve the privacy of the data in multiple participating parties.

6 Conclusion

Skyline queries are an important type of multi-criteria analysis with diverse applications in practice. In this paper, we adopt the classical honest-but-curious attack model and present a suite of efficient protocols for skyline queries to answer skyline queries in a privacy-preserving way when the data is sensitive and distributedly owned by multiple parties. The secure protocols for horizontally partitioned data take 1-to-N skyline dominance as the primitive secure operation to prevent the revelation of intermediate results. We analyze in detail the efficiency of each protocol and its privacy guarantee.

References

1. Abdel Wahab, O., Hachami, M.O., Zaffari, A., Vivas, M., Dagher, G.G.: Darm: A privacy-preserving approach for distributed association rules mining on horizontally-partitioned data. In: IDEAS (2014)
2. Amirbekyan, A., Estivill-Castro, V.: A new efficient privacy-preserving scalar product protocol. In: AusDM (2007)
3. Atallah, M.J., Du, W.: Secure multi-party computational geometry. In: Intl Workshop on Algorithms and Data Structures (2001)
4. Ben-David, A., Nisan, N., Pinkas, B.: Fairplaymp: A system for secure multi-party computation. In: CCS (2008)

5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC (1988)
6. Blanton, M., Aguiar, E.: Private and oblivious set and multiset operations. In: ASIACCS (2012)
7. Bogdanov, D., Laur, S., Willemson, J.: Sharemind: A framework for fast privacy-preserving computations. In: ESORICS (2008)
8. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: ICDE (2001)
9. Burkhart, M., Strasser, M., Many, D., Dimitropoulos, X.: Sepia: Privacy-preserving aggregation of multi-domain network events and statistics. In: USENIX Security (2010)
10. Cachin, C.: Efficient private bidding and auctions with an oblivious third party. In: CCS (1999)
11. Chaum, D., Crépeau, C., Damgard, I.: Multiparty unconditionally secure protocols. In: STOC (1988)
12. Du, W., Atallah, M.J.: Protocols for secure remote database access with approximate matching. In: E-Commerce Security and Privacy (2001)
13. Fagin, R., Naor, M., Winkler, P.: Comparing information without leaking it. Commun. ACM 39(5) (1996)
14. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC (1987)
15. Gordon, D.S., Carmit, H., Katz, J., Lindell, Y.: Complete fairness in secure two-party computation. In: STOC (2008)
16. Harnik, D., Naor, M., Reingold, O., Rosen, A.: Completeness in two-party secure computation: A computational view. In: STOC (2004)
17. Holzer, A., Franz, M., Katzenbeisser, S., Veith, H.: Secure two-party computations in ansi c. In: CCS (2012)
18. Huang, Y., Evans, D., Katz, J., Malka, L.: Faster secure two-party computation using garbled circuits. In: USENIX Security Symposium (2011)
19. Ibrahim, M.H.: Two-party private vector dominance: The all-or-nothing deal. In: ITNG. pp. 166–171 (2006)
20. Jin Yuan, Qingsong Ye, H.W., Pieprzyk, J.: Secure computation of the vector dominance problem. In: ISPEC (2008)
21. Laur, S., Talviste, R., Willemson, J.: From oblivious aes to efficient and secure database join in the multiparty setting. In: ACNS (2013)
22. Liu, C., Wang, X.S., Nayak, K., Huang, Y., Shi, E.: Oblivm: A programming framework for secure computation. In: IEEE Symposium on Security and Privacy (2015)
23. Michael J. Freedman, K.N., Pinkas, B.: Efficient private matching and set intersection. In: EUROCRYPT (2004)
24. Mihir Bellare, Viet Tung Hoang, S.K., Rogaway, P.: Efficient garbling from a fixed-key blockcipher. In: IEEE Symposium on Security and Privacy (2013)
25. Miyaji, A., Rahman, M.S.: Privacy-preserving data mining: A game-theoretic approach. In: DBSec (2011)
26. Naor, M., Pinkas, B.: Oblivious transfer and polynomial evaluation. In: STOC (1999)
27. Naor, M., Pinkas, B.: Efficient oblivious transfer protocols. In: SODA (2001)
28. Vaidya, J., Clifton, C.: Privacy-preserving top-k queries. In: ICDE (2005)
29. Vaidya, J., Kantarcioglu, M., Clifton, C.: Privacy-preserving naive bayes classification. The VLDB Journal 17(4) (2008)
30. Yao, A.C.: Protocols for secure computations (extended abstract). In: FOCS (1982)