eSkyline: Processing Skyline Queries over Encrypted Data*

Suvarna Bothe[1] Panagiotis Karras[1] Akrivi Vlachou[2]
[1]Rutgers, The State University of New Jersey [2]NTNU, Trondheim, Norway

ABSTRACT

The advent of cloud computing redefines the traditional query processing paradigm. Whereas computational overhead and memory constraints become less prohibitive, data privacy, security, and confidentiality concerns become top priorities. In particular, as data owners outsource the management of their data to service providers, query processing over such data has more resources to tap into, yet the data oftentimes has to be encrypted so as to prevent unauthorized access. The challenge that arises in such a setting is to devise an encryption scheme that still allows for query results to be efficiently computed using the encrypted data values. An important type of query that raises unconventional requirements in terms of the operator that has to be evaluated is the skyline query, which returns a set of objects in a dataset whose values are not dominated by any other object therein. In this demonstration, we present *eSkyline*, a prototype system and query interface that enables the processing of skyline queries over encrypted data, even *without* preserving the order on each attribute as order-preserving encryption would do. Our system comprises of an encryption scheme that facilitates the evaluation of domination relationships, hence allows for state-of-the-art skyline processing algorithms to be used. The actual data values are reconstructed only at the client side, where the encryption key is known. Our demo visualizes the details of the encryption scheme, allows a user to interact with a server, and showcases the efficiency of computing skyline queries and decrypting the results.

1. INTRODUCTION

Consider a database $\mathcal{DB}$ whose tuples are represented as a set of multidimensional data points. A point p then belongs to the *skyline* set $\mathcal{S}_{sky}$ if there is no other point p' that can improve on p in at least one dimension without worsening in any other dimension. Such points collectively represent the best tradeoffs among the different aspects of the data. The problem of identifying such skyline points has been known in economics as the Pareto optimum problem and in optimization theory as the maximum vector problem [10].

*Work supported by a Rutgers Business School RRC grant.

Proceedings of the VLDB Endowment, Vol. 6, No. 12

Skyline query processing attracts consistent attention in database research, due to its applications in decision making and analytics [4, 11, 8]. For example, assume database $\mathcal{DB}$ contains numerical information about hotels in the Maldives. The attribute values of hotel h (e.g., price per night, distance to beach) can then define the coordinates of the point p representing h. A user may be interested in hotels that are both inexpensive and close to the beach. However, it is not clear how such a user would assess a hotel very close to the beach but more pricey than others, or a relatively cheap hotel farther away from the beach. In other words, a scoring function based on which we could assess and compare different hotels based on the relative importance of their attributes is not available or not applicable. The skyline query avails us of the need for such a scoring function, as it simply retrieves all hotels that cannot be matched or bettered in both price and distance to the beach by any other. Figure 1 presents an example of such data, where the coordinates of each point stands for a hotel's attributes. The skyline set is $\{a, i, m, k\}$.

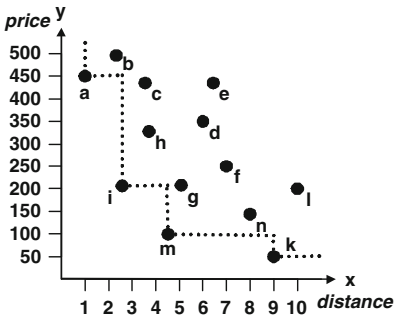

Figure 1: Skyline Example

Apart from the discussed hotel booking scenario, skyline queries find application in electronic marketplaces. For instance, in air travel sales, a user may wish to navigate the Pareto tradeoff created by the price, number of stops, and flight duration, while a user buying a car may wish to see which cars are provide a good tradeoff in the criteria of fuel consumption, age, and price.

Skyline query processing has hitherto been studied in either centralized or distributed environments, where one or more data vendors dispose their actual data in one or more servers, and these data are then processed in order to acquire the skyline set. Nevertheless, nowadays data vendors have financial and technological incentives to exploit emerging computing paradigms such as database outsourcing, content sharing, and cloud computing, in which the task of data processing it taken over by a remote service provider. In this scenario, additional security and confidentiality concerns enter the picture, as the remote service provider may be relied upon to provide computing infrastructure, but not so when it comes to disclosing private customer data and business information [6]. For

instance, airlines, travel agencies, and hotel groups may wish the data on their offers to be available for querying and comparison shopping at a remote server, but not readily available for download by their competitors; similarly, a national statistics bureau may wish to allow querying over confidential census data for the purpose of determining outlier and skyline values, but not for the sake of making the full corpus of such data available to a third party. Under these circumstances, the need arises to devise query processing techniques that can operate over data that has been encrypted so as not be exposed to untrusted third parties, including the service provider. In this framework, a service provider is called to process queries securely over the encrypted data without accessing the data's real values [9]. Thus, the employed encryption scheme should support the execution of queries over the encrypted data. Past research has examined how point queries, range queries, and nearest-neighbor search can be performed under such confidentiality constraints [12, 14]. However, no method has hitherto been proposed that achieves the same effect for skyline queries.

In this demonstration, we present *eSkyline*, a system and interface that encrypts database tuples and efficiently processes skyline queries over them; it employs a scalar-product-preserving encryption scheme that allows for domination relationships among encrypted tuples to be "blindly" determined by the server without the plain data being revealed. It does so without disclosing the order among attribute values, as an Order-Preserving Encryption Scheme [1] would do, and without carrying the computational burden of fully homomorphic encryption [7]. After a skyline result set has been reported, real data values can be decrypted and reconstructed at the client side via a set of transformation matrices. The functionalities of our model can be incorporated into a system of database outsourcing for the particular aim of processing skyline queries.

2. TECHNICAL OVERVIEW

In this section we discuss the main building blocks of our system. We start out by presenting skyline computation techniques, which we adopt from previous research for our purposes. Then we outline the novel encryption scheme we employ. Last, we present the architecture of the prototype system we have implemented for use in this demonstration. We emphasize that an analysis of the security features of our encryption scheme with regards to the computational power of an adversary falls outside the scope of this demo paper and is relegated to a more extensive research article.

2.1 Skyline Computation

Symbol	Description
m	Dimensionality
$\mathcal{DB}$	Dataset
t, s	Tuples of $\mathcal{DB}$
$\mathcal{S}_{sky}$	Skyline tuples

Table 1: Overview of Notations.

We are given a dataset $\mathcal{DB}$ on a data space defined by a set of m attributes (dimensions) $\{d_1, ..., d_m\}$. A tuple $t \in \mathcal{DB}$ is represented as $t = \{t_1, ..., t_m\}$ where t_i is the value on dimension d_i. Without loss of generality, we assume that $\forall d_i : t_i \geq 0$, and that larger values of each attribute are preferable to smaller ones. Table 1 provides an overview of the notation we employ.

DEFINITION 1. **Skyline Query** *We say that a tuple $t \in \mathcal{DB}$* dominates *another tuple $s \in \mathcal{DB}$, as relationship denoted as $t \prec s$, if (1) on every dimension d_i, $t_i \geq s_i$; and (2) on at least one dimension d_j, $t_j > s_j$. The* skyline *is the set of points $\mathcal{S}_{sky}$ that are not dominated by any other point in $\mathcal{DB}$.*

In the example of Figure 1, the x-dimension represents a hotel's distance to a point of interest such as the beach, while the y-dimension indicates the price of a room in the hotel. Accordingly, a hotel dominates another hotel when it is both cheaper and closer to the beach (or equal on one aspect and better on the other). Thus, the skyline points a, i, m and k represent the most attractive tradeoffs between price and distance to the beach that are available.

Regardless whether the data set $\mathcal{DB}$ is encrypted or not, the computation of a skyline requires a capacity to determine one fundamental relationship among any pair of tuples $\{s, t\}$: to ascertain whether one dominates the other. In addition, it is useful to possess the values of any function $f(t)$ that is *monotonic* on each attribute t_i of t, i.e., non-decreasing as the value of t_i changes in the direction of higher preferability. Under the assumption that larger values are preferable to smaller ones, we can simply use the *aggregate* function over all attribute values as our $f(t)$. We can then efficiently compute the skyline set using the Sort-Filter-Skyline (SFS) algorithm of previous research [5], shown in Algorithm 1.

Algorithm 1 Sort-Filter-Skyline

```
1: Input: data set DB sorted by f(t)
2: Output: skyline set of DB
3: S_sky ← ∅
4: for (each tuple t in DB) do
5:    dominant = TRUE;
6:    s = first point in S_sky;
7:    while ((dominant) ∧ (s ≠ NULL)) do
8:       if s dominates t then
9:          dominant = FALSE;
10:      end if
11:      s = next point in S_sky;
12:   end while
13:   if dominant then
14:      S_sky = S_sky ∪ {t};
15:   end if
16: end for
17: return S_sky
```

2.2 Encryption Mechanism

To our knowledge, the only extant solution for secure skyline querying over some form of encrypted data is provided in [15]. This solution uses OPES, the Order-Preserving Encryption scheme proposed in [1]. Unfortunately, OPES, built on an encoding that preserves the order of the numerical data in each column, is not sufficiently secure; an adversary who observes the encrypted data effectively learns the order of tuples on each attribute, which is a significant amount of information by itself [13, 2, 3].

Nevertheless, as we show, order preservation per se is not a necessary condition to enable the processing of skyline queries over encrypted data. After all, the fundamental operation we have to perform in order to answer a skyline query is the operation that decides whether a tuple t dominates or is dominated by another tuple s. In other words, given a pair of tuples in $\mathcal{DB}$, (t, s), we need to decide whether there is a *domination relationship* among them, as in Line 8 of Algorithm 1. Furthermore, the problem of determining domination, as a decision problem, does not need to specify whether it is t that dominates s or s that dominates t; it suffices to ascertain whether a domination exists among them. The specification of the dominant and dominated tuples can be easily performed as an auxiliary operation, provided that the values of monotonic function f for both tuples are available; the tuple of higher f values is the dominant one. Thus, without loss of generality, for each

tuple t we can always compute such a function as the *sum* of all attribute values, and treat it as attribute t_0.

Then, assuming each tuple has m (original) attributes, and using "$\cdot$" to denote the scalar product of vectors, the existence of a domination relationship between s and t can be expressed as follows:

$$(t_i - s_i) \cdot (t_{i+1} - s_{i+1}) > 0, \quad 0 \leq i \leq m-1 \tag{1}$$

For the sake of conciseness, we use $>$ in our inequalities, without prejudice to the fact that the strict domination definition requires a $>$ inequality along at least one dimension, while it suffices to have a $\geq$ inequality along other dimensions. In our implementation, we take care of the strict definition of domination. For each tuple vector $t =< t_0, t_2, \ldots, t_m >$, we construct a set of sub-vectors:

$$t^j =< t_0, t_2, \ldots, t_j >, \quad {}^j t =< t_1, ..., t_{j+1} >, \quad 0 \leq j \leq m-1$$

Then the set of Inequalities (1) can be also expressed as:

$$(t^0 - s^0) \cdot ({}^0 t - {}^0 s) > 0$$
$$(t^j - s^j) \cdot ({}^j t - {}^j s) - (t^{j-1} - s^{j-1}) \cdot ({}^{j-1} t - {}^{j-1} s) > 0 \tag{2}$$

For $1 \leq j \leq m-1$. Following the exact definition of domination, we can express the condition as follows: The series of terms

$$(t^j - s^j) \cdot ({}^j t - {}^j s), \quad 0 \leq j \leq m-1 \tag{3}$$

should be an *non-descending series* of non-negative terms, with at least one pair of consecutive terms being strictly *ascending*.

In effect, the problem of determining domination between two tuples is mapped to a problem of computing scalar products among the sub-vectors of tuples, as well as comparing their f values, in case domination exists. We can then deal directly with the differences among these sub-vectors and the scalar products of those. Let $p = t - s$. Then it suffices to compute terms of the form:

$$p^j \cdot {}^j p \tag{4}$$

We should then have a method for computing scalar products of this form, where p may have been computed from any two tuples in $\mathcal{DB}$. Such scalar products can in turn be expressed as:

$$\Big((p^j)^T \times M_j\Big) \times \big(M_j^{-1} \times {}^j p\Big) \tag{5}$$

Where p^j and ${}^j p$ are represented as column vectors, M_j is any invertible $(j+1)\times(j+1)$ matrix, "$\times$" denotes matrix multiplication, and a^T denotes the transpose of a. Based on the above analysis, if we have a set of $m-1$ invertible matrices $M_1, M_2, ..., M_{m-1}$, we can encrypt all the sub-vectors we need as follows:

$$E(t^j) = M_j^T \times t^j, \quad E({}^j t) = M_j^{-1} \times {}^j t \tag{6}$$

where $E()$ is our encryption function. Then, for any t, s, with p calculated as $p = t - s$, and for any $j \in [0, m-1]$, it holds that $E(p^j) \cdot E({}^j p) = (M_j^T \times p^j)^T \times M_j^{-1} \times {}^j p = (p^j)^T \times {}^j p = p^j \cdot {}^j p$. Thus, all scalar product calculations we need can be effectively performed by working solely over encrypted sub-vectors and their differences. At same time, given the encrypted sub-vectors of a tuple t, $E(t^j)$ and $E({}^j t)$, it is not possible for an adversary to determine the value of t_j or any other metric of t without knowing M_j. In other words, the set of invertible matrices is the key of our encryption scheme.

We emphasize that the attribute t_0 of any tuple t undergoes a relatively trivial encryption, as matrix M_0 is only a scalar value. However, this feature does *not* constitute a weakness of our scheme. On the contrary, we intentionally design our scheme this way so that values of attribute t_0, namely the monotonic function $f(t)$, allow their *order* relationships to be recovered after being encrypted. That is, as long as M_0 is positive, their order remains unchanged. On the other hand, it takes significantly more effort to recover the order, or, for that matter, the values of other attributes, apart from the extent to which those can be inferred from domination relationships.[1]

Using the encryption scheme, we can safely compute whether there is a tuple s dominates another tuples t. In more detail, given a data set $\mathcal{DB}$ of m-dimensional tuples and a set of m invertible matrices $M_0, M_2, ..., M_{m-1}$, as described above, and a monotonic function $f(t)$, the *encrypted* from of $\mathcal{DB}$, $\mathcal{DB}'$, is created by mapping each tuple $t \in \mathcal{DB}$ to a $m^2 + m$-dimensional tuple t' formed by concatenating the contents of t^j and ${}^j t$ for $0 \leq j \leq m-1$. Thereafter, a service provider holding the encrypted data applies the SFS algorithm directly on encrypted tuples in order to answer skyline queries. The only modification required is that now domination is ascertained by computing a sequence of scalar products. For each domination check, this sequence of scalar products is aborted as soon as a product gives negative result, which indicates that domination does not hold. After all calculations, the server returns the encrypted form of tuples to the client. An encrypted sub-vector $E({}^{m-1} t)$ of t suffices to decrypt all of t's attributes' values at the client side by the matrix-vector multiplication $t = M_j \times {}^{m-1} t$.

2.3 System Architecture

We have implemented a prototype system that performs the tasks of both server and client. A database $\mathcal{DB}$ is encrypted using a series of randomly generated matrices M_j and uploaded to the server module, while observing user preferences during encryption. Thereafter, the server processes user-issued skyline queries over the encrypted data and return the results. These results are decrypted at the client side and the final result presented to the user. Figure 2 presents a screenshot of the user interface during the final result decryption, along with a visualization of the final skyline tuples via a colored bar-gram, with one bar per attribute per tuple. We present some additional details of our system architecture in the following.

Environment The encrypted database is loaded on IBM DB2 Version 9.7, while skyline computation is implemented in C++. Windows forms were used to create user interfaces. Testing experiments were conducted on a Microsoft Windows 2007 workstation with a 1.60GHz Intel Core i7 processor and 6GB of memory.

Data Set We have obtained data sets from the IPUMS archive[2]. Our data consists of a relation of 10^5 records containing demographic information about economic characteristic of US households, including attributes such as cost of Electricity, Gas, Water, and Fuel. The number of attributes loaded an encrypted can be adjusted by the user. During encryption, a monotonic function $f(t)$ is calculated based on user's preferences and appended at the beginning of each tuple t as attribute t_0. The data type used for this first attribute is decimal, so as to allow sorting by value. All other attributes are stored as strings. The database is accessed in the front end through the ODBC driver that is available with IBM DB2, while the ODBC Data Reader is used for reading the records.

Encryption Stage Prior to encryption, a user is asked to specify a preference for each attribute, i.e., whether lower or higher value is preferable. The system then calculates the $f(t)$ function, stored as attribute t_0, and produces the elongated vector representing each tuple. The incorporation of t_0 serves two purposes: (i) it does not allow the actual first attribute to be easily inferred, and (ii) it allows for tuples to be sorted so that the SFS algorithm can be applied.

[1] An adversary could launch a known plaintext attack in order to determine that contents of matrices M_j; still, a complete security analysis of our scheme is outside the scope of this demo paper.

[2] Downloaded from www.ipums.org

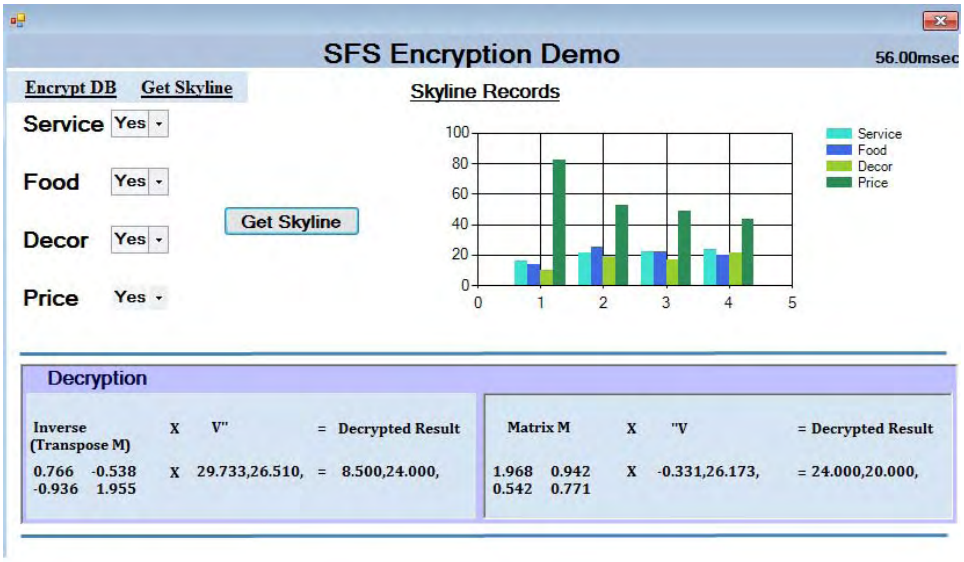

Figure 2: A Screenshot of the eSkyline interface

Thereafter, a series of invertible random square matrices, M_j, are generated. The elements of these matrices are constrained in the range $[0.2, 5]$ so as to delimit error propagation during vector-matrix multiplication. These matrices constitute our encryption key; they are shared only at the client side. These matrices are then employed to generate the contents of the encrypted tuples by Equation (6) and insert them in the encrypted database.

Skyline Query Processing Skyline computation is performed for a specific subset of attributes selected by the user. Our SFS implementation caters to subspaces and computes domination relationships by taking into account only the attributes in a selected subspace. This effect is achieved by omitting some scalar product computations and catering only to those for relevant attributes.

Decryption Stage Once the skyline result $\mathcal{S}_{sky}$ is obtained, the encrypted tuples are returned to the client. At the client side, each skyline tuple t is decrypted as $t = M_{m-1} \times^{m-1} t$. For verification, other sub-vectors can also be decrypted using the respective matrices, using the inverse of the matrix employed for encryption. Sample matrices used in the decryption stage and a decrypted skyline result for a simple data set are depicted in Figure 2.

3. DEMONSTRATION EXPERIENCE

Our demonstration showcases the functionality of the *eSkyline* system. The audience can witness all the internal workings of encryption and decryption, as well as issue skyline queries over the database instance we have extracted from IPUMS. Visitors will have the option of determining, via the user interface, whether a lower or higher value of each numerical attribute is preferable. Following the selected settings, they system goes though the encryption stage that produces encrypted tuples to be used at the server side. Sample encrypted values are displayed in a data grid.

Visitors can navigate through the encrypted database. They can (i) issue skyline queries by selecting a subset of attributes of interest, and (ii) select and upload new data to the server, using the same encryption scheme. Through interactive exploration, participants will be able to witness how the skyline in each subspace is affected after new tuples are added, how skyline of different subspaces differ from each other, while all are obtained over the encrypted data. Moreover, the user interface presents the time it takes to compute a skyline over the encrypted data (upper right corner in Figure 2); the audience will also have the choice to execute the same query over the non-encrypted data, so as to witness the affordable time overhead required to work over the encrypted database, and confirm that the system's interactivity remains unfettered by encryption.

4. CONCLUSION

This demonstration paper presented *eSkyline*, a novel system architecture for processing skyline queries over encrypted data. Our system allows for a service provider to answer queries posed by clients without having the actual data values disclosed. This functionality is enabled by a novel encryption scheme we proposed, with allows for the domination among two tuples to be efficiently determined, yet does *not* use any form of order-preserving encryption. In the future, we aim to validate our results with very large data sets following diverse value distributions, and expand our solution to the case where the data distributed among several servers.

5. REFERENCES

[1] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *SIGMOD*, 2004.

[2] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, 2009.

[3] A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011.

[4] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *ICDE*, 2001.

[5] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In *ICDE*, 2003.

[6] F. Farahmand. Risk perception and trust in cloud. *ISACA Journal*, 4, 2010.

[7] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, 2009.

[8] P. Godfrey, R. Shipley, and J. Gryz. Maximal vector computation in large data sets. In *VLDB*, 2005.

[9] H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *SIGMOD*, 2002.

[10] K. Hose and A. Vlachou. Distributed skyline processing: a trend in database research still going strong. In *EDBT*, 2012.

[11] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In *SIGMOD*, 2003.

[12] W. K. Wong, D. W.-L. Cheung, B. Kao, and N. Mamoulis. Secure kNN computation on encrypted databases. In *SIGMOD*, 2009.

[13] Z. Yang, S. Zhong, and R. N. Wright. Privacy-preserving queries on encrypted data. In *ESORICS*, 2006.

[14] M. L. Yiu, G. Ghinita, C. S. Jensen, and P. Kalnis. Enabling search services on outsourced private spatial data. *The VLDB Journal*, 19(3):363–384, 2010.

[15] T. Zenginler. Secure skyline querying. Master's thesis, Boğaziçi University, 2007.