

# SECURE SKYLINE QUERYING

by

Tansel Zenginler

B.S. in Computer Engineering, Boğaziçi University, 2003

Submitted to the Institute for Graduate Studies in  
Science and Engineering in partial fulfillment of  
the requirements for the degree of  
Master of Science

Graduate Program in Department of Computer Engineering  
Boğaziçi University

2007

SECURE SKYLINE QUERYING

APPROVED BY:

Prof. Taflan Gündem .....  
(Thesis Supervisor)  
  
Prof. Fikret Gürgen .....  
  
Assoc. Prof. Birgül Kutlu .....

DATE OF APPROVAL: 14.09.2007

## ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor Prof. Taflan Gündem, for being not only such a great mentor, but also a great person with his kind, never-ending encouragement and nice talks. I have been really fortunate to have the chance of working with him. I would also like to thank to approvers of my thesis, Fikret Gürgen and Birgül Kutlu, for their encouragements.

I would like to express my sincere thanks to all of the the faculty staff of Computer Engineering Department of Boğaziçi University for tolerating and encouraging me during my thesis study. They guided me with their experience. I am really grateful to them for their interest, concern, and valuable counsel on both research and writing of this thesis. They spent a great deal of time especially in dealing with my LaTeX and other matters. This thesis would certainly not come to an end without them.

I wish to express my gratitude to the greatest parents, Nurettin and Şükran Zenginler and my sister Tuğba Zenginler ever for embracing me with their endless love and care. My appreciation for them is beyond the words for making me feel the most precious creature on Earth. They are the most effective supporters for this thesis, the ones who wanted me to end this thesis with success.

Also, I would like to thank to people and friends in Beko Electronics. Their encouragement and belief on me are very helpful in my work for thesis. They tolerate me very much in the thesis phase of my master program. It can be harder if I could not get their help. By means of their help, I can show that it can be possible to get along work and school at the same time.

This thesis is completed thanks to the encouragements of all my friends. I would like to thank them for opening the great doors of academic enthusiasm and giving very nice lectures on life. They were always with me and amused me a lot with their nice and unique comments on everything. Checking out my papers, listening to my

presentations, tolerating me and entertaining me with their humorous talks are their assistants near me. Besides, they help me very much by making me the happiest person in the world.

Last but not least, special thanks to those anonymous people who once touched my soul.

## ABSTRACT

### SECURE SKYLINE QUERYING

The skyline of a set of  $d$ -dimensional points contains the points that are not dominated by any other point on these  $d$ -dimensions. Querying skyline points and its computation has recently received considerable attention in the database community. Skyline points are important in database systems. They are useful for multi-criteria decision making systems and data mining systems. There are several algorithms that query data by Skyline computation. Also, there are several algorithms that query requested data from encrypted data. In my proposed system, the Skyline computation is proposed in a secure way. I combine encryption process and skyline computation in a single system. For encryption, order preserving encryption, OPES is used, which helps in the performance. The security of the system is strengthened by the schema hiding procedure in the proposed system. By this system, the skyline computation can be outsourced in a secure way. The client-server architecture is used for outsourcing. In summary, this system proposes Skyline computation in a secure way.

## ÖZET

### GÜVENLİ SKYLINE SORGULAMALARI

Boyutu  $d$  olan noktalar kümesi üzerindeki Skyline işlemi verilen  $d$  boyut için hiç bir nokta tarafından domine edilmeyen noktaları içerir. Skyline sorgulamaları ve hesaplanması son zamanlarda veritabanı ile ilgilenen topluluklar tarafından büyük ilgi görmüştür. Skyline noktaları veritabanı sistemlerinde önem taşımaktadır. Bu noktalar çok kriterli karar verme sistemleri ve veri madenciliği sistemlerinde çok kullanışlıdır. Günümüzde Skyline hesaplamaları ile veri sorgulamaları yapan algoritmalar mevcuttur. Bunun yanında şifrelenmiş veri üzerinden sorgulama yapan algoritmalar da mevcuttur. Sunulan sistemde, Skyline hesaplamaları güvenli bir yoldan yapılmıştır. Şifreleme süreci Skyline hesaplamaları ile entegre edilmiştir. Şifreleme için, sıralamayı koruyan şifreleme metodu olan ve böylece performansa katkısı bulunan OPES kullanılmıştır. Sunulan sistemde, sistemin güvenliği şema saklama prosedürü ile güçlendirilmiştir. Bu sistem ile Skyline hesaplamalarında güvenli bir yol ile dış kaynak kullanılabilir. Sunucu-istemci mimarisi dış kaynak kullanımı için kullanılabilir. Özet olarak, bu sistem Skyline hesaplamalarını güvenli bir yolla yapmamızı sağlar.

## TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . . .	iii
ABSTRACT . . . . .	v
ÖZET . . . . .	vi
LIST OF FIGURES . . . . .	ix
LIST OF TABLES . . . . .	xi
LIST OF SYMBOLS/ABBREVIATIONS . . . . .	xii
1. INTRODUCTION . . . . .	1
2. PRELIMINARIES . . . . .	5
2.1. Skyline . . . . .	5
2.2. Security . . . . .	9
2.3. Client-Server Architecture . . . . .	12
3. PROPOSED SYSTEM . . . . .	14
3.1. Main Architecture . . . . .	14
3.2. Security of the System . . . . .	17
3.2.1. Order Preserving Encryption . . . . .	17
3.2.2. Meta Data Hiding . . . . .	18
3.3. Skyline of the System . . . . .	20
3.4. Overall Algorithm . . . . .	23
4. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS . . . . .	25
4.1. Simulation Setup and Scenarios . . . . .	25
4.1.1. Hardware Requirements . . . . .	25
4.1.2. Software Requirements . . . . .	25
4.1.3. Simulations . . . . .	26
4.2. Performance Evaluation . . . . .	27
5. CONCLUSIONS . . . . .	30
APPENDIX A: ALGORITHMS . . . . .	32
A.1. Skyline Algorithms . . . . .	32
A.1.1. BNL . . . . .	32
A.1.2. SFS . . . . .	33

A.1.3. DC . . . . .	33
A.1.4. Epsilon Skyline . . . . .	34
A.1.5. Bitmap-based Skyline . . . . .	34
A.1.6. Index-based Skyline . . . . .	34
REFERENCES . . . . .	35



## LIST OF FIGURES

Figure 1.1.	Skyline of the hotel dataset . . . . .	2
Figure 2.1.	Constrained Skyline Query . . . . .	7
Figure 2.2.	DC Method in Skyline . . . . .	8
Figure 2.3.	Encryption Methods . . . . .	10
Figure 2.4.	Client Server Architecture . . . . .	12
Figure 3.1.	Client Server Architecture . . . . .	14
Figure 3.2.	Algorithm at the client side . . . . .	16
Figure 3.3.	Algorithm at the server side . . . . .	16
Figure 3.4.	Dividing Attributes into Subattributes . . . . .	18
Figure 3.5.	Data After Encryption and Property Conversion data at client side	19
Figure 3.6.	Data View at the server side . . . . .	20
Figure 3.7.	First Step for Skyline . . . . .	21
Figure 3.8.	Second Step for Skyline . . . . .	21
Figure 3.9.	Last Step for Skyline . . . . .	22
Figure 3.10.	Algorithm of the overall system . . . . .	23

Figure 4.1.	Encrypted Format of the Hotel Dataset . . . . .	26
Figure 4.2.	Property Data Stored at client side . . . . .	27
Figure 4.3.	Properties stored at server side . . . . .	27
Figure 4.4.	Data Stored at server side . . . . .	28

## LIST OF TABLES

Table 1.1.	Example dataset of an hotel database . . . . .	1
Table 2.1.	Subspace Skyline . . . . .	7

## LIST OF SYMBOLS/ABBREVIATIONS

BNL	Block Nested Loop
DC	Divide and Conquer
SFS	Sort Filter Skyline
OPES	Order Preserving Encryption Scheme
IO	Input-Output

## 1. INTRODUCTION

In recent years, database and information retrieval research communities concentrated on a topic named Skyline. Computation of skyline queries is very important in database systems. The importance of skyline points in applications such as multi-criteria decision making systems, data-mining systems and user-preference queries attract research communities. A skyline point is the one that is not dominated by any other points over  $d$  dimensions and a skyline query over  $d$  dimensions selects the skyline points. The search criteria determines the dominance among objects. Domination means the situation that if an object  $a$  is as good or better in all dimensions and better in at least one dimension than another object  $b$ ,  $a$  dominates  $b$ . There has to be at least one dimension to be better than another object [1], [2].

Table 1.1. Example dataset of an hotel database

	Price(\$)	Distance(meters)
<b>A</b>	90	100
<b>B</b>	100	200
<b>C</b>	80	400
<b>D</b>	70	600
<b>E</b>	100	900
<b>F</b>	50	700
<b>G</b>	60	500
<b>H</b>	30	400
<b>I</b>	20	300
<b>K</b>	10	900
<b>L</b>	40	1000
<b>M</b>	20	600
<b>N</b>	30	800

As mentioned previously, in multi-criteria decision making systems, data-mining systems and user-preference queries, skyline querying and its result sets are important. finding hotel problem problem is the most popular problem queried with skyline. It is a

multi-criteria decision making problem to find a hotel with nearer distance to the beach and a lower price. To give an example of that problem, hotel A with distance=100 meters and price=\$90 dominates hotel B with distance=200 meters and price=\$100. Because hotel A has lower price and nearer to the beach than hotel B. However, it is impossible to find hotels near to the beach and cheaper than others. The hotel near to the beach tend to be more expensive. In such a complementary goal situation, finding the best choice may be difficult.

As in the example of hotel database, the complementary goal situation makes the database systems unable to decide the best choice. For example, in the dataset of a travel agent's as in Table 1.1, we can unable to find the best hotel. However we can at least present all interesting hotels. The set of interesting hotels is called Skyline. The user can choose the best hotel from the non-dominated hotel set of skyline points. By Skyline querying we can eliminate all dominated hotel records. The graphical representation of Table 1.1 and Skyline points of that dataset is given in Figure 1.1.

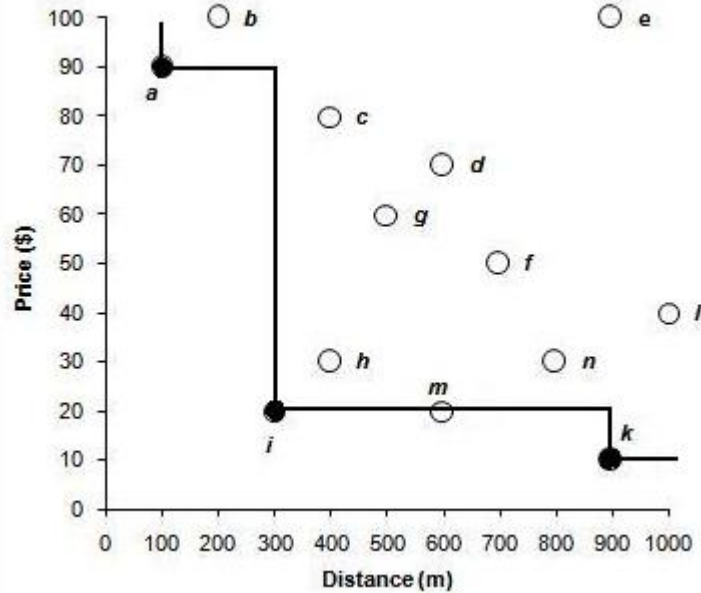


Figure 1.1. Skyline of the hotel dataset

The researches about database systems in recent years has concentrated another topic, security. The security of database systems is important because of existence of sensitive data. The privacy of data is a requirement for Hippocratic databases [3],

military databases, datasets that contain personal and private data. One of the mechanisms that is used for protecting privacy is access control. Access control mechanisms are necessary for database systems. However, the way of accessing raw files of database systems makes these systems insufficient.

Extra systems are needed to protect data. Encryption is one of the established techniques to protect sensitive data. To protect sensitive data by encryption we use some resources for it. Because of this usage, some performance degradation arises. This degradation in performance makes integration of encryption mechanisms to the database systems be drawback. If we need a query that requires full table scan, we have to decrypt the data to get the query. This means we need to decrypt all the table to get the query operations. Besides, we have indices (such as B-Tree) by which we can query faster. However, to encrypt and decrypt data makes the system to lose indices. Because of these reasons, an encryption schema that preserves order is developed [4]. By this algorithm, order of the data is not corrupted. The comparison operations can be done without the need of decryption. These encryption techniques can be seen as order preserving encryptions [4]. The performance degradation in encrypted systems decreases by using such algorithms for encryption.

In spite of not to get the plaintext from the encrypted data, an attacker can get some statistical information about private data. The statistical information about the data space may give a user the chance of identify the data. Especially for order preserving encryption schema, an attacker can get minimum or maximum objects, can compare objects without the need of decryption. To prevent an attacker to get this statistical information from the encrypted data, database systems need extra protection systems. To hide the meta data of the dataset is such approaches. To give an example, we may protect attributes of the data or structure of the data. If an attacker can access the way to decrypt the data, he/she can get the data in decrypted format. But he/she does not know about the structure of data. Also, the probability of getting statistical information from the decrypted data is decreased by meta data hiding.

The skyline query has many implementations and usage areas. Also, security is

very important in database systems. In this thesis, we will show skyline queries in a secure way. The usage of order preserving encryption with an extra protection of meta data hiding for skyline queries makes the system secure. Besides the security, the performance issue for skyline queries is also considered. The skyline methodologies are studied to get the best fit skyline query methodology for that system.



## 2. PRELIMINARIES

In multi-criteria decision making systems, data-mining systems and user-preference queries, skyline querying and its result sets are important. Because of this importance, Skyline querying received considerable attention in recent years.

In addition to Skyline querying, security and privacy of sensitive data attract attention in computer science and in mathematics in recent years. This is due to the sensitivity of private data in database systems. The privacy of data in Hippocratic databases and in some other databases is in our lives. Because of this importance, security received a lot of attention. Also the performance issue in security takes attention in last years.

In the next two sections, the summary about how and which work done on them will be given.

### 2.1. Skyline

In database systems, we may have complementary goal situations. In such a case, there is not a best result. If we want a goal to be best, we lose from other goals. The best choice over these goals become a list of objects. In a set of  $d$ -dimensional objects, the best objects that are not dominated by others are called skyline objects. The skyline objects are returned as a result of a skyline query. In other words skyline query over  $d$ -dimensions returns the objects that are not dominated by other objects. The calculation of domination is determined in skyline by search criteria. The domination of an object to another object means the object that dominates is as good or better in all dimensions and better in at least one dimension. The most used example of skyline queries is the hotel dataset example. The aim of that skyline query is to find the hotel with nearest to the beach and lowest price as in Table 1.1 and Figure 1.1. This hotel example is a multi-criteria decision making problem. However as mentioned before, these two goals are complementary. The hotel nearer to the beach tend to be

more expensive than others. This situation makes to find a hotel with both minimum distance and lowest price almost impossible. Hence, finding the best choice may be difficult for such a case [5].

There are many works done on skyline, its query types, operators, performance and storage structure. In general, the "MIN" operator is used. By this operator, the dominant objects are chosen from the dataset by using "MIN" function in mathematics. The value that is smaller than the other is more valuable or dominant. For example, the hotels that are nearer to the beach are dominant to the others. This means that the distance from the hotel to the beach must be smaller to be dominant as in Table 1.1 and Figure 1.1 [1], [2].

In addition to "MIN" operator in skyline, there are some other operation types for skyline querying. For example, we can see "MAX" function as an opposite of "MIN" function. In this case, an object is dominant if the value is greater than the others. These functions can be seen as monotone functions or only comparison operations. The order of the objects is more important for such functions than the values to query [5].

Range queries in skyline is another research area in database communities. This type of operations in skyline queries can be called as Constrained skyline queries [6]. In the constrained skyline query, not all of the objects is considered. Objects that do not meet the constrained, are not considered in dataset. We may think this dataset as a sub dataset of the original one as in Figure 2.1.

Skyline queries traditionally is interested with fixed number of dimensions, the number of dimensionality of the dataset. However in real life, users may be interested with different attributes of objects or different subsets of attributes. The price and distance to the beach of the hotel is important for one person whereas the other one is interested with the number of rooms, quality and the price. In addition to these difference in the dimensionality for the queries, the system must handle with high dimensionality. To query on all dimensions can be seen as full space skyline. Besides, considering only some dimensions on the data can be named as subspace skyline. By

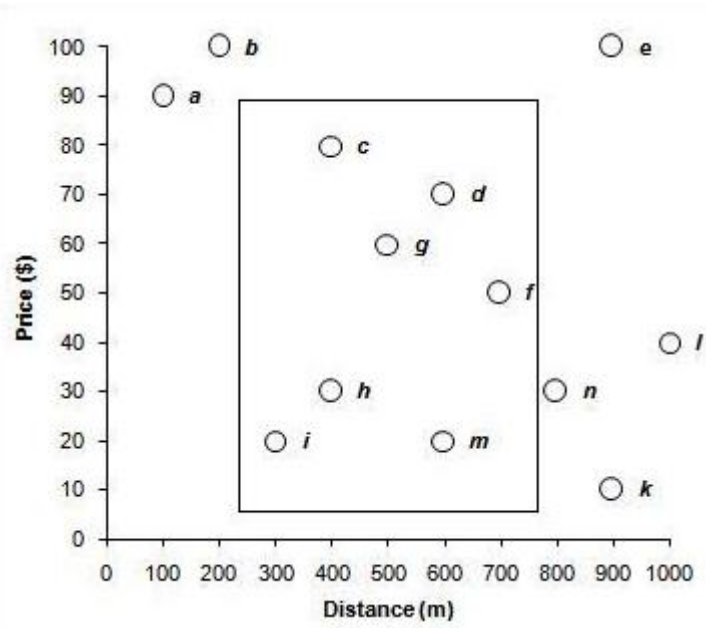


Figure 2.1. Constrained Skyline Query

using subspace skyline queries, an operator can be defined by which only the objects of interested properties are returned to the user. Subspace skyline is proposed recently in [5], [6], [7], [8]. The data sample and its subspace queries are given at Figure 1.1 and Table 2.1 respectively.

Table 2.1. Subspace Skyline

	Subspace	Skyline
1	Distance	a
2	Price	k
3	Distance, Price	a,i,k

In stead of subspace skyline, there may be some other kind of operator, such as constrained skyline query. This operator can return the objects in the given range. Also subspace skyline and constrained skyline queries may be combined as in [6].

The concept improved after the development of subspace skyline is SkyCube. SkyCube is the complete computation of all possible subspace skylines for all dimensionality of the full space. By this precalculation procedure of skyline, the system stores the results for all subspace skylines. This may be thought as OLAP in database

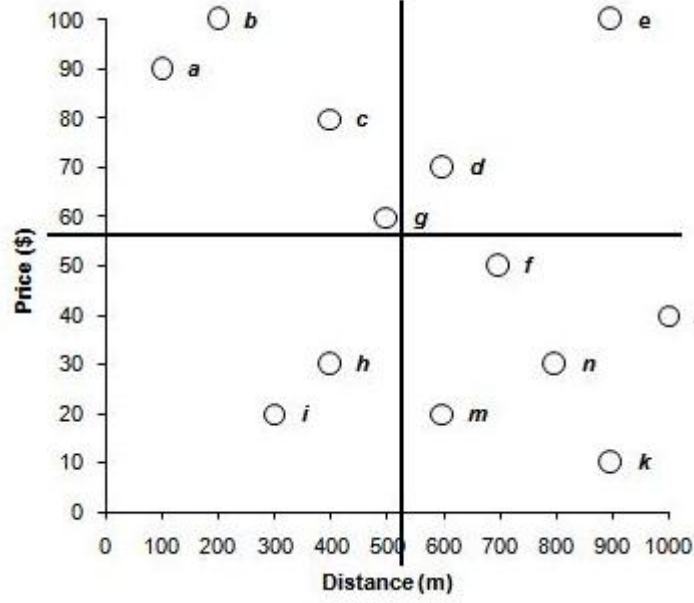


Figure 2.2. DC Method in Skyline

systems. In this approach, the system has to store  $2^d-1$  skyline results where  $d$  is the dimensionality of the whole dataset [5], [7], [9], [10], [11], [12]. The system has to give on-the-fly results to all of the subspace skylines by the improvement of SkyCube.

Compression is another hot topic for skyline querying. Especially as a result of the development of skycube, the more space is required. This more space requirement makes the system inefficient for both storage and performance. Because of these new approaches, compression for skyline and SkyCube is studied in recent years [5], [13]. By compressing the SkyCube, the intersections of skylines are stored only ones, and a skyline of a subspace can be computed by the assistance of the other subspace skylines.

In addition to the studies about the operators and methodologies of Skyline, some other researches done on continuity [14] and scalability [15], parallelization [16] of the Skyline query. Query types [17], dominance relationships [18], [19], and the domains used in skyline querying [20] are other kinds of recently studied topics on Skyline Querying.

The computation and methodologies of skyline querying are other research areas

in addition to the different applications developed for skyline [21]. Divide and Conquer (DC), Block Nested Loop (BNL), Sort Filter Skyline (SFS), Bitmap-Based Skyline and Index-Based Skyline querying are some of these methodologies [1], [2], [9], [22]. BNL compares each point of the database with every other point. It reports the point as a result only if it is not dominated by any other point. The BNL approach is the most general for all problems and the simplest way for skyline. DC in Figure 2.2 divides the data space into several regions. It calculates the skyline in each region, and produces the final skyline from the union of points in the regional skylines. SFS, which is based on the same principle as BNL, improves performance by first sorting the data according to a monotone function [23]. As a result of this presortion step, an object dominates the ones after it. Besides of these studies, index-based techniques are proposed in several works. Two of them are Bitmap and Index method in which there is no need to scan the whole dataset. In another work, a branch and bound algorithm is proposed [1], [2]. It progressively outputs skyline points of a dataset indexed by an R-Tree. This technique guarantees the minimum I/O cost for a given R-tree.

## 2.2. Security

The progress in computer technologies, such as networking, processor architecture, and storage has attached more importance to digital data. Private and sensitive data exists in the electronic format. The amount of that data increases in recent years. The security of that data has become one of the most important issues in database and information retrieval communities. For example, for Hippocratic databases [3], [24], [25], military databases, and datasets that contain personal and private data [26], [27], [28], [29] are such databases.

The first and simplest method that we could implement to protect the sensitive data is access control. By this approach, users can access the data source in an authorized way. However, to be able to get access to raw data files makes this protection method insufficient.

The insufficient properties of access control mechanisms have forced new protec-

tion systematics to be developed for sensitive and private data. Encryption is one of the established techniques to protect sensitive data. Encryption is the process of transforming information (referred to as plaintext) to make it unreadable to anyone. Only the people possessing special knowledge, usually referred to as a key, can access the plaintext. The reverse process is the decryption process. Decryption is to make the encrypted information readable again (to make it decrypted). In cryptography, a cipher (or cypher) is an algorithm for performing encryption and decryption a series of well-defined steps that can be followed as a procedure 2.3.

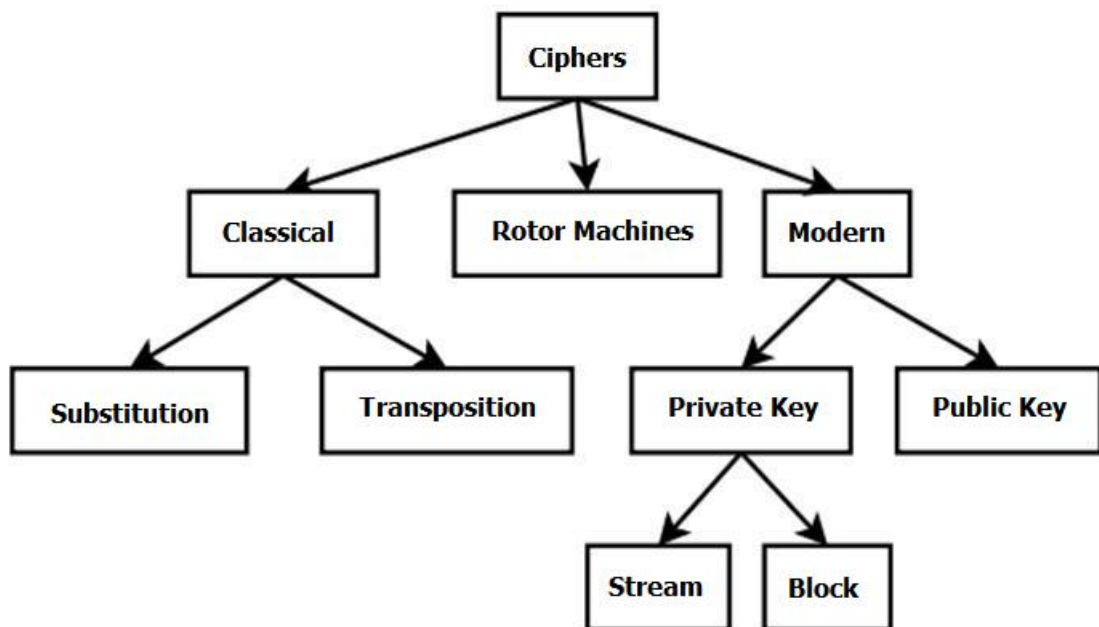


Figure 2.3. Encryption Methods

In classical techniques, substitution and transposition is used for encryption. The process in those methodologies is to change a plaintext with another one. The rotor machines makes this procedure in a physical environment. In modern systems, the key is used for encryption and decryption. According to broadcasting the key, it is public or private.

We can protect sensitive data better by encryption. However, there is an extra resource usage for encryption process. This usage causes to a performance degradation. Integration of encryption mechanisms to the database systems bothers the system. In

addition to encrypt data and usage of resources at the encryption stage, we may also need to decrypt at query stage. We may need a query that requires full table scan and the query operator needs the decrypted data directly. In such a case, we can get the query by only decrypting the whole data. By encryption and decryption processes we also lose indices (such as B-Tree) of that table. By such indices, we can query faster. We need to use an encryption schema that does not force user to decrypt to query data. We can gain some of these performance degradation by using such encryption schemas. For example, we need a query that needs to compare data by a monotone function such as minimum or maximum operations in mathematics. We can get the result without decrypting if we use an order preserving encryption schema. Because, the order of the objects are preserved. And, the query only needs the order to get the result. It only needs to compare the objects. These encryption techniques can be seen as order preserving encryptions [4].

In order preserving encryption represented in [4], there are three phases to be used, Model Phase, Flatten Phase and Transform Phase. There is another concept, called buckets, used in OPES. The data is fit to the buckets determined in the model phase. In the first phase, model phase, the distributions and bucket boundaries are determined. Bucket boundaries are determined in two phases, the growth phase and prune phase. In the growth phase, the buckets that are too large are divided into sub-buckets until the number of objects in a bucket is below some threshold. In the prune phase of modeling phase, the buckets are tried to be combined to decrease the sum of the cost of describing the system. The next phase is the flatten phase in which the plaintext buckets is mapped into buckets in flatten space. In this phase, there are two parameters used to flatten data, mapping function and scale factor. By these parameters, we could flatten the data. The total of parameters, mapping functions and scale factors for each bucket, formes the encryption key. In the transform phase, the last phase of OPES, flattened buckets is transformed into the target distribution. The target distribution is determined by the user. By the order preserving encryption schema, the order of the data is preserved. The queries that only needs the order of the data to compare each data can use the encrypted data without the need of decryption. Besides, while preserving the order, OPES get the data sorted after encryption [4].

Encryption makes attackers not to get the real values from systems. However, there is no prevention of getting statistical information by using encryption. Especially, if the order preserving encryption schema is used, an attacker has more chance to get more statistical data or to identify the data. The attacker can not know the real values but he/she can know which object is greater or smaller than the other. Meta data hiding can be seen as an extra prevention schema in addition to encryption. By using meta data hiding, the structure of the dataset can be protected from an attacker more, in addition to get statistical information from the decrypted data [30].

### 2.3. Client-Server Architecture

The recently popular trend in computer systems is to concentrate on own business instead of dealing with other additional tasks, such as the security or hardware issues. Outsourcing is the way to concentrate on your own business. The user can outsource the software that he uses, the data that he queries, the hardware that he uses. For example, the database of the system can be outsourced to make the security, storage systems and other requirements of the system to be handled by the other side of the outsourcing. Figure 2.4 shows the basic visualization of the client server architecture. The client can be seen as the requester of services whereas the server as the provider of services. The client uses the services provided by the server. The storage systems, cpu power or the security of the server is used by the client.

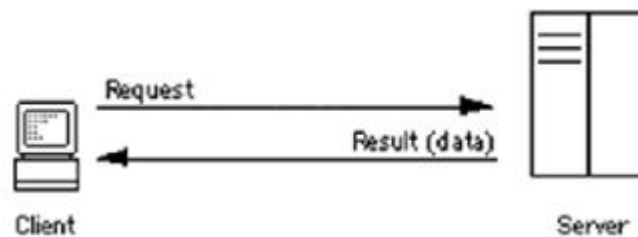


Figure 2.4. Client Server Architecture

The Client-Server architecture has some advantages and disadvantages. The main advantage of this architecture is to use the resources of the server by the clients. Storage system, cpu power, ram or security system of the server can be used by client queries. By outsourcing, all the data can be stored on server side. However, in such a case, the



security becomes a problem to be handled. Besides, network congestion is an important factor for querying time. In addition, if the server fails, the clients that need to that server, can not get query results.

### 3. PROPOSED SYSTEM

In preceding chapters, security of database systems and a new approach, skyline querying, are mentioned in a general manner. In this thesis, the security and skyline querying are combined. In the following sections, I give details of combining order preserving encryption, OPES, meta data hiding and skyline querying. First of all, the main architecture of the proposed system is given in the following section. The security of the system is mentioned after the main architecture section. The security of the system is studied on two subsections, the encryption section and meta data hiding section. After those details, the skyline algorithm on this proposed system is given. The final section of that chapter is the overall algorithm of the system.

#### 3.1. Main Architecture

The main issue in this thesis is to get skyline query in a outsourced and secure way. The outsourcing is done by client-server architecture, and the security is preserved by encryption and meta data hiding. The details of security preservation will be given in the following section. This section has the detail of outsourcing and main architecture of the proposed system.

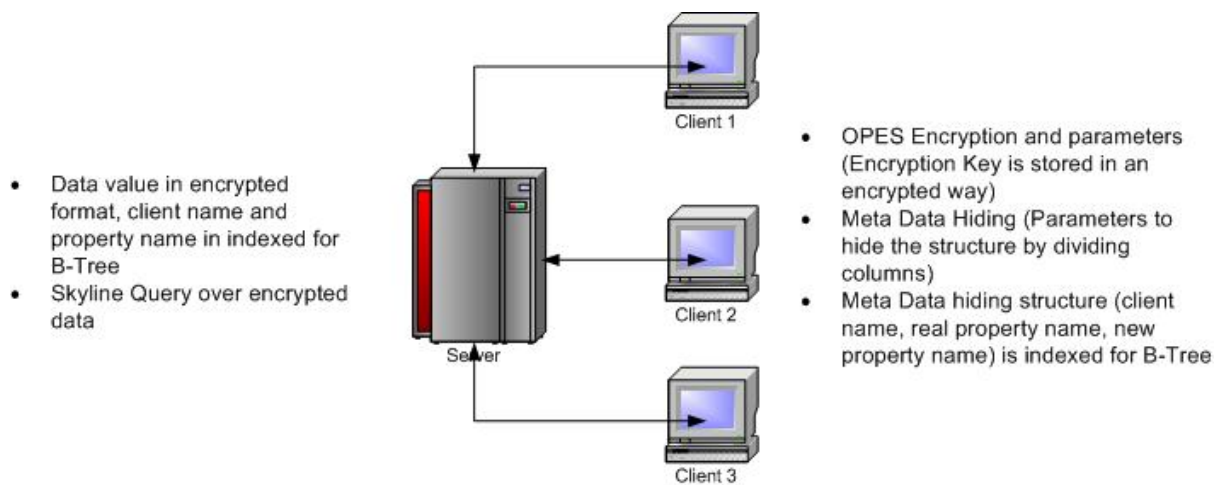


Figure 3.1. Client Server Architecture

In the client-server architecture of this system, there are two node types, the

client and the server. The client is responsible for sending data to server, asking for queries, encryption of the data and hiding the meta data. The server does not know anything about the work done on client side. The storage of encrypted data, skyline of requested queries and sending them to the client is the work of server.

The data at the client side is organized as follows. The client gets the data in plaintext. The plaintext contains objects to be queried by skyline. All objects in the data have one identifier field. This identifier property, called index, is the first property of objects. The other properties come after the index. They are the properties that we could query by skyline. All of the sensitive properties, except from the identifier field, have to be encrypted according to order preserving encryption. This encryption process is also done at the client side. The order preserving encryption parameters are stored at the client side as encrypted. The properties of data can be seen as the dimensionality of the system. If a data has two properties except from the identifier property, then this data is said to be a 2-dimensional data.

In the meta data hiding process, a property is divided into  $n$  buckets. All properties are divided into same number of buckets. The client stores a property conversion data. This data contains the client name, old property name and the property identifier we get after conversion. The conversion means that we divide the range of the property into smaller buckets and give new identifier numbers to hide the property name. This conversion data has also minimum and maximum values of that bucket. The meta data hiding parameters are indexed according to client name, table name, property name, and minimum, maximum values. By this index, we can search an attribute by B-Tree and find the new property name that is used in server. Besides every new property has minimum value and maximum value to be used in skyline operations. The usage of these values are explained later in the step of getting skyline query at the server side. All these processes in client side, summarized in Figure 3.2 are mentioned in the meta data hiding section of the security section in detail.

In the client-server architecture of the proposed system, the data is stored and queried at the server side. The server stores the data in encrypted format. It does

Client	
PreProcess	1 Encrypt Data
	1.1 Encryption with OPES
	2 Metadata hiding
	2.1 Divide each dimension with parameter n
	2.2 Store metadata hiding parameters (Tablename, propertyname) with index
	3 Send data to server by webservice
Query	1 Get query parameters
	2 Recalculate query parameters (Metadata Hiding)
	3 Send parameters to server
	4 Get result and go to step 3 until of end of parameters (Check if the rest of the regions are dominated by any point from that result)
	5 Combine results with BNL Skyline
	6 Decrypt combined result
	7 Show result

Figure 3.2. Algorithm at the client side

not know anything about the values of the data and the structure of the database. The server only stores the data in the encrypted format. The server can not decrypt data, because the server does not have any information about the encryption key. The encryption keys are stored at the client side. The server also does not know about the structure of the database, it only stores the information about client name, property name and the value (in encrypted format) and the index of the value. This is the only information at the server side. These data is organized at the server side as follows. In a table the client names and property names are stored with an index over both of them. By this index, we can search the requested data by B-Tree. We get the client name, property name by this B-Tree effectively. We get the index number of the the requested property of the requester (the client name) by this B-Tree.

Server	
Preprocess	1 Get data from client
	1.1 Webservice invoked
	2 Store Data with path index
	2.1 Store ClientName, PropertyName parameters with index
	2.2 Store data with the above index and sorted values
Query	1 Get query parameters
	2 Skyline with SFS
	3 Send result to client

Figure 3.3. Algorithm at the server side

As a result of the search by B-Tree of client name and property name, we get the index of that client and property. We can find the data of that client and property by that index. The data is in encrypted format. This search is also by B-Tree because of the index over the previous index number found and the data. Also the data is sorted in descending order, as a result of OPES. This process in server side, summarized in Figure 3.3 is mentioned in detail in the Skyline of the System section.

### 3.2. Security of the System

The security of the outsourced data is provided by order preserving encryption and meta data hiding. One of them provides the values of the sensitive data, another is for hiding the structure. The following subsections give details about these processes.

#### 3.2.1. Order Preserving Encryption

In this thesis, the skyline system with outsourcing data to the server needs security concerns to be established. This security need can be partially covered by encryption. The skyline query needs comparison operations to get the skyline points. The point that is not dominated by any other point is a skyline point. The domination is determined by the search criteria. Most used skyline operator is the "MIN" operator in mathematics. According to this operator in skyline querying, a value which is lower than others, it is more likely for that object to dominate others in skyline query [5]. The general characteristic of the operators that are used in skyline query is being monotone functions and "MIN" operator is one of these operators. The skyline query needs to compare objects according to these operators. As a result of this characteristic, the skyline operators interest with the order of the objects instead of the real values.

The order of the objects to get the non-dominated objects by skyline is more important than the value itself. Because of this usage of the order of the data, the security process have to not to change the order of the data. Any change in the order of the data makes the system to need decryption. The preservation of the order for that system can be obtained by encryption methods that preserves the order. We can

use any order preserving encryption schema that preserves order to get skyline query at the server without the need of decryption. OPES is one of this kind of encryptions mentioned in [4]. OPES can be used at the client side without any change. However, we simulate the encryption part as input and output, the OPES algorithm as a black box and if we give the input, the output is given in the encrypted format. Because of the non-changed usage of OPES, we can get its complexity as mentioned in [4].

### 3.2.2. Meta Data Hiding

Real values, which means the plaintext values, are not known by any user other than the client. The client stores the encryption keys in the encrypted format. But, there are two risks, the attacker can get access to the encryption keys or he/she get statistical information about the data. For example, the maximum or minimum objects can be found because of the usage of the order preserving encryption. However, for the security of the private and sensitive data, no statistical information has to be accessed [30]. To prevent this kind of information from the attackers, we must hide the structure and meta data from the servers and attackers, which is called meta data hiding in this context.

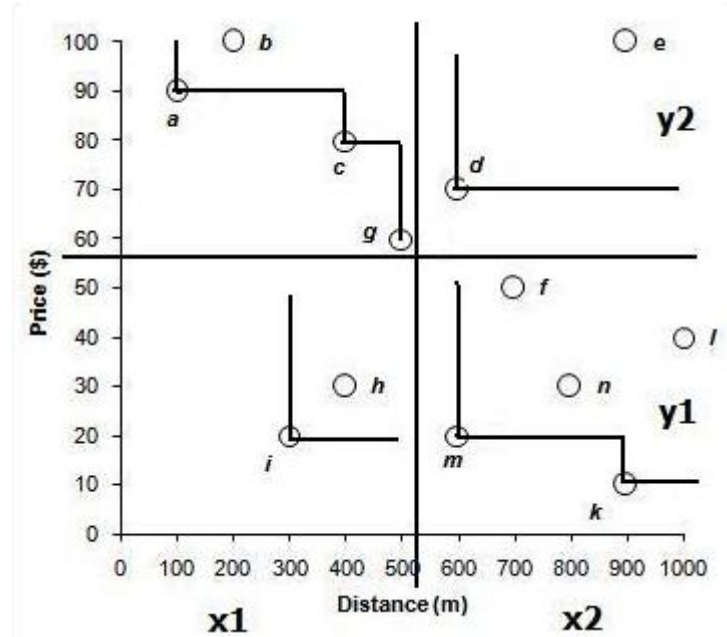


Figure 3.4. Dividing Attributes into Subattributes

The first step to hide the meta data is to divide the attributes (properties) or columns of the database to the sub-attributes or sub-columns. As in Table 3.4, axis X is divided into two sub-properties x1 and x2, Y is divided into also two, y1 and y2. The new attribute names called x1, x2, y1 and y2 in example, however, the system generates an incremental number for that names in the system as 1 for x1, 2 for x2, 3 for y1 and 4 for y2.

Id	X	Y
1	1	9
2	2	10
3	4	8
4	6	7
5	9	10
6	7	5
7	5	6
8	4	3
9	3	2
10	9	1
11	10	4
12	6	2
13	8	3

ID	CLIENT_NAME	TABLE_NAME	PROPERTY_NAME	MIN_VALUE	MAX_VALUE
10	Tansel	Hotel_Data	X	1	6
9	Tansel	Hotel_Data	X	6	10
12	Tansel	Hotel_Data	Y	1	5
11	Tansel	Hotel_Data	Y	6	10

Figure 3.5. Data After Encryption and Property Conversion data at client side

The server does not know anything about X and Y. It knows that an object in the data has value for property 1, 2, 3 or 4. This property identifier is sent by the client to the server. In the example given in Figure 3.5 and Figure 3.6, the server side and client side data is given. The client side has data in encrypted format and a conversion parameter to change the parameter names to the new identity numbers. For example, the first object, with id 1, in the encrypted data has value 1 for X and 9 for Y. Value 1 for X is in the range of the first bucket of X, id with 10. The client converts the data of the first object in the following way. The object with id 1 has a property named 10 and value 1. The same procedure is done for Y and the other properties if there exists any. As a result of this process, the server has the data as in Figure 3.6. There is no information about the properties of the table. The server has information about the table name and it thinks that there exists 4 properties of that table.

CLIENT_NAME	PROPERTY	ID	DATA_INDEX	DATA_VALUE
Tansel	9	269	11	10
Tansel	9	267	10	9
Tansel	9	268	5	9
Tansel	9	266	13	8
Tansel	9	265	6	7
Tansel	9	264	4	6
Tansel	10	276	12	6
Tansel	10	275	7	5
Tansel	10	273	8	4
Tansel	10	274	3	4
Tansel	10	272	9	3
Tansel	10	271	2	2
Tansel	10	270	1	1
Tansel	11	281	5	10
Tansel	11	282	2	10
Tansel	11	280	1	9
Tansel	11	279	3	8
Tansel	11	278	4	7
Tansel	11	277	7	6
Tansel	12	289	6	5
Tansel	12	288	11	4
Tansel	12	286	13	3
Tansel	12	287	8	3
Tansel	12	284	12	2
Tansel	12	285	9	2
Tansel	12	283	10	1

Figure 3.6. Data View at the server side

### 3.3. Skyline of the System

If a client wants a skyline query from the server, firstly it has to decide the attribute names to want from the server. For example, if the client needs a skyline query over the dimensions X and Y as in Figure 1.1, the client initially has to know the new attribute names, x1, x2, y1 and y2 as in Figure 3.4. This information is stored at the client side with the index and B-Tree over table name and property name. In other words, client knows that X dimension is divided into x1 and x2 and it has an index over these information to get these with B-Tree.

The first step to get the skyline after getting the new attribute information is to get the skyline query for the region which has the lowest values of all dimensions as in region 1 of Figure 3.7. The most important region for the whole dataset is region 1. If a point in that region is a skyline object, then objects in region 2's and 3 in Figures 3.8 and 3.9 may be dominated by that point. If a point in region 1 is returned as a skyline object from the first query request from the server, this point is looked for its



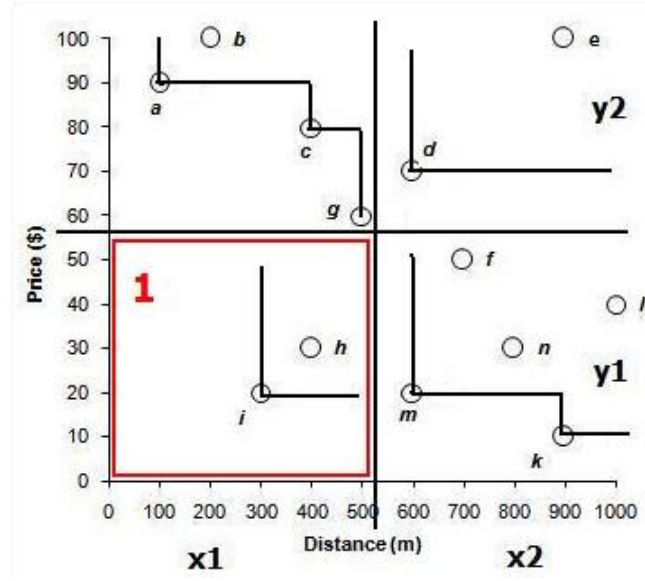


Figure 3.7. First Step for Skyline

domination of other regions. If that point dominates the lower left corner (the object with minimum values of that region) any other region, then the query for that region is not needed. Every point at that region is dominated by that skyline point. By this part, we get skyline query of the whole system more effectively.

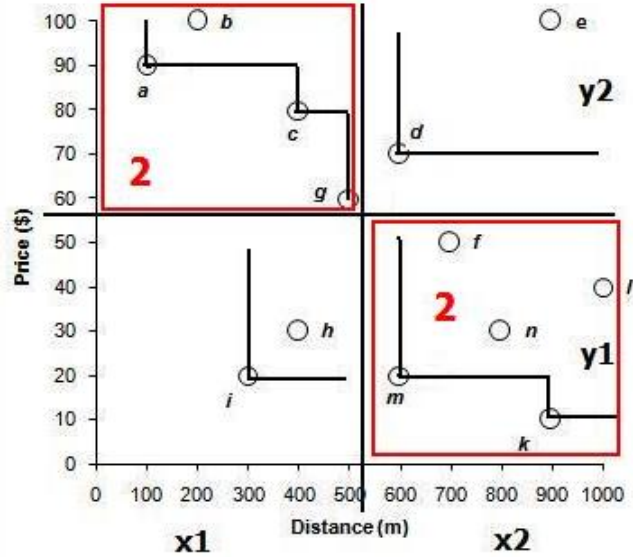


Figure 3.8. Second Step for Skyline

In the examples of Figures 3.7, 3.8 and 3.9, the skyline point  $i$  in region 1 of Figure 3.7 dominates region 3 of Figure 3.9. However, region 2's of Figure 3.8 are not dominated by the point  $i$ . The domination of a region by a point is determined

by comparing the lower left corner of that region with recently found skyline points. If one of the skyline points dominates the lowest point of a region, this means that skyline point can dominate every other point at that region. By this checking procedure, we eliminate the skyline query for region 3. This makes the system has better performance. However, this procedure gives the client an extra load to check if the point dominates a region or not.

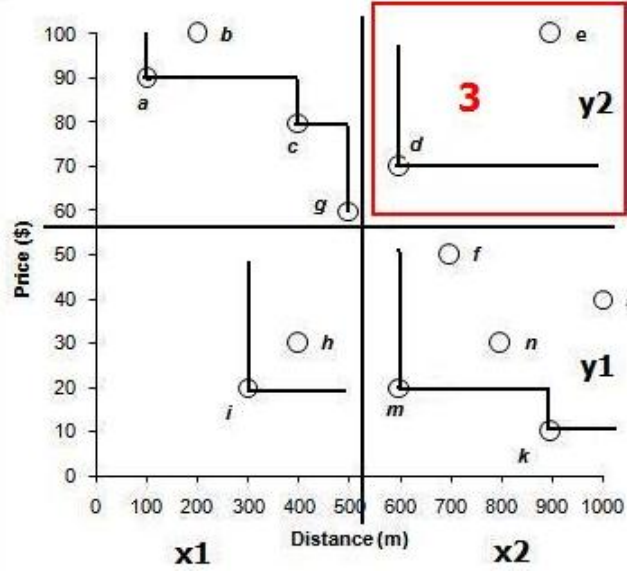


Figure 3.9. Last Step for Skyline

As can be seen from the Figure 3.8, region 2's are not dominated by the point  $i$  of region 1. Hence, two regions marked with 2 should be queried from the server. However region 3 of Figure 3.9 is dominated by point  $i$  and this query is not needed.

The last step at the client side is to merge the results which the client get partially from the server. In this example, it has to merge skyline points of regions 1 and 2's. After this merge, we get the result from that skyline query. That skyline query at the client side is BNL, because there is no sorted data and there are too few data. Hence, we get the query with the most general method of Skyline at client side. The client uses BNL Skyline, whereas the server side uses SFS approach of Skyline. Because data is in sorted format as a result of OPES at the server side. In addition, we use B-Tree as the server side, to get faster searches. However, in the SFS method of server side, there is no need pre-sorting step because of the already sorted and indexed data as

mentioned. This makes the server has less operations, and get the advantages of SFS algorithm without sorting.

### 3.4. Overall Algorithm

The overall algorithm of the system has two parts, the preprocess step and query step as in Figure 3.10. The preprocess step includes the procedures of data gathering, encrypting, meta data hiding and sending this data to server. The Query step has the procedures of getting which parameter sets are queried from the server, getting these results, checking if to go on with the other regions and if finished combining results and showing the overall result after decryption. The row by row steps of the algorithm is given in Figure 3.10.

	Client	Server
PreProcess	1 Encrypt Data	
	1.1 Encryption with OPES	
	2 Metadata hiding	
	2.1 Divide each dimension with parameter n	
	2.2 Store metadata hiding parameters (Tablename, propertyname) with index	
	3 Send data to server by webservice	
		4 Get data from client
		4.1 Webservice invoked
		5 Store Data with path index
		5.1 Store ClientName, PropertyName parameters with index
Query		5.2 values
	1 Get query parameters	
	2 Recalculate query parameters (Metadata Hiding)	
	3 Send parameters to server	
		4 Get query parameters
		5 Skyline with SFS
		6 Send result to client
	7 Get result and go to step 3 until of end of parameters (Check if the rest of the regions are dominated by any point from that result)	
	8 Combine results with BNL Skyline	
	9 Decrypt combined result	
	10 Show result	

Figure 3.10. Algorithm of the overall system

The overall system can be seen as formed by two sides, the client and the server. Both sides have procedures for both preprocess step and query step. The client is responsible for encryption, meta data hiding and sending the data to the server at the

preprocess step. In the preprocess step, the server has to get the data and store it according to B-Tree. The other step for both sides are query step. The first thing to do is to arrange the new property names given after the meta data hiding procedure. According the arrangement, the client asks for queries. The most important gain of the proposed algorithm is to check the regions if needed or not. By this approach, the client may not need to query some regions.

## **4. EXPERIMENTAL RESULTS AND PERFORMANCE ANALYSIS**

### **4.1. Simulation Setup and Scenarios**

#### **4.1.1. Hardware Requirements**

The simulations are run over a computer with the hardware of Intel Centrino Duo 1.66GHz, 2Gb Ram, 80Gb 5400Rpm Harddisk. Both the client and server are simulated at the same machine at the same time.

#### **4.1.2. Software Requirements**

The system was written in Microsoft Visual Studio 2005 in a computer with operating system Windows XP with Service Pack 2.

The Client is a web program run over IIS service of Windows operating system. It is an ASP.NET web application over .NET Framework 2.0 and coded using C#, edited in Visual Studio 2005.

The server is a web service coded with C#. This webservice is communicate with the client program over XML which makes the system has the ability to work with other clients coded over other platforms.

The data is over Microsoft SQL Express server 2005 for both the client and server sides. The indexes are created over this platform and therefore B-Tree is used for searching data.

### 4.1.3. Simulations

The initial data is at the client side for simulations. The data at the client side is to be encrypted and to be sent to the server. In the basic implementation of this system, this initial data is in the SQL server. The first column is the ID field, which is the identity number of the objects. The other attributes are the attributes over which we can get Skyline query. The attributes other than ID, are to be encrypted by OPES.

The basic implementation of this system has a web site for clients. In this web page, there are 3 choices, upload data to the server, get attributes of a client to get query and the last but not least query the system from the server. These 3 choices are performed by 3 buttons on the web site.

Simulations of that system was driven over two datasets, one of a 2-dimensional hotel database, and the other with 18-dimensional 16380 NBA Player records.

The encrypted format of the dataset of the hotel database example is given in Figure 4.1.

Id	X	Y
1	1	9
2	2	10
3	4	8
4	6	7
5	9	10
6	7	5
7	5	6
8	4	3
9	3	2
10	9	1
11	10	4
12	6	2
13	8	3

Figure 4.1. Encrypted Format of the Hotel Dataset

The example corresponding table formed at the client side and used for property conversion is given in Figure 4.2. This table stores the client name, the table name and property values.

ID	CLIENT_NAME	TABLE_NAME	PROPERTY_NAME	MIN_VALUE	MAX_VALUE
10	Tansel	Hotel_Data	X	1	6
9	Tansel	Hotel_Data	X	6	10
12	Tansel	Hotel_Data	Y	1	5
11	Tansel	Hotel_Data	Y	6	10

Figure 4.2. Property Data Stored at client side

The server side information is given below figures, Figure 4.3 and Figure 4.4.

ID	CLIENT_NAME	PROPERTY
64	Tansel	9
65	Tansel	10
66	Tansel	11
67	Tansel	12

Figure 4.3. Properties stored at server side

To give an example of the hotel database, we can consider the object with ID 1. This object has value 1 for X and 9 for Y after encryption. Value 1 for X property lies in the first bucket of X, with the ID 10, as in Figure 4.2. This object is sent to the server as an object with ID 1 and has value 9 for property 10. The server gets this information and gets skyline according to that information. The client has to know that it needs to query property 10 and 9 for X.

The system is simulated as a 2-dimensional database with 2 buckets for each property in the previous example. It is suitable for any dataset with any dimension number and any bucket size.

## 4.2. Performance Evaluation

The proposed system has some performance issues. We can think this issue in three parts. The encryption part, meta data hiding procedure and the query stage at the server.

In this thesis, we used order preserving encryption, OPES, to encrypt data. The input and output of this process is simulated in the system. The performance issue for

ID	CLIENT_PROPERTY_ID	DATA_INDEX	DATA_VALUE
269	64	11	10
267	64	10	9
268	64	5	9
266	64	13	8
265	64	6	7
264	64	4	6
276	65	12	6
275	65	7	5
273	65	8	4
274	65	3	4
272	65	9	3
271	65	2	2
270	65	1	1
281	66	5	10
282	66	2	10
280	66	1	9
279	66	3	8
278	66	4	7
277	66	7	6
289	67	6	5
288	67	11	4
286	67	13	3
287	67	8	3
284	67	12	2
285	67	9	2
283	67	10	1

Figure 4.4. Data Stored at server side

that part can be seen from [4].

The skyline method SFS is at the server side. SFS is used because of the usage of sorted data in it. We also have sorted data as a result of OPES. Hence, we do not need the presortion step at SFS. As a result, this is a modified version of SFS, without sorting step. By this view, the performance issue of SFS algorithm is used as same in this approach. We only need to get out the part of sorting step.

The BNL algorithm is used at the client side to merge the results of the subregions. The data is not sorted here, and the cardinality of the data is low. Hence, we used the most general approach of skyline. We also need to add its performance to get the overall performance of the system.

The most important performance issue lies in the meta data hiding step. The overall dataset is divided into subregions at this stage. Both the dimensionality of the system and the number of buckets are important for the whole system. The more we



bucketize the system, we may need more queries from the server. However, the distribution of the system is also important for the performance. We can take two examples for that issue, one with uniform distribution and one with non-uniform distribution after encryption. In the uniform distribution, we only need the query regions 1 and 2's in Figures 3.7, 3.8 and 3.9. There is no need to query for region 3. To generalize this, we have to query  $((d*n)-1)/n^d$  queries, where  $n$  is the number of buckets and  $d$  is the dimensionality. In the worst case, the all data is non-uniformly distributed such that all data is in the region 4 in Figure 3.9. In this case, we need to query all regions. As a result, we have  $((d*n)-1)/n^d$  for best case, and the same result as SFS for the worst case. In an example, if we have a uniform distribution, and have 2 buckets for each two dimensions, we have  $((2*2)-1)/2^2$  for performance. This means we gain 25 percent from the performance.

However, to hide the dimensionality and statistical data, meta data hiding is done which makes the server returns more results than real results. We need an extra query to get the correct result.

As a result of meta data hiding, less data compared in each query because of partitioned data space, like DC method of Skyline. This means we query less data with SFS. However, the usage of B-Tree makes the system perform well.

## 5. CONCLUSIONS

In this thesis, we studied on security and outsourcing for skyline. The client-server architecture is used for outsourcing. Encryption and meta data hiding are used for security.

In the client-server architecture of the proposed system, the client handles with the encryption and meta data hiding steps, whereas the server is responsible for storing data and skyline querying with SFS. After the server returns the results of the queries. Client combines the results by BNL Skyline and shows the final result after decryption.

Encryption is provided by Order Preserving Encryption Schema, OPES. Meta data hiding is another part for security, which seems as DC method of Skyline. By this security procedures, we can outsource the data, and use the client-server architecture to load the cpu and memory burden processes to the server. Also, the server is responsible for storing data. By encryption and meta data hiding, an attacker or the server does not know anything about the data. The attacker also does not know anything about the statistical information about the data. By these procedures, the outsourced skyline querying can be used for Hippocratic databases, banking system (eg. Risk vs Scale of the firm) or military databases (eg. Distance vs Bomb-Target Meeting Percentage).

Data storage and skyline querying are done on server side; whereas encryption-decryption, attribute partitioning, and schema storage on client side. The most of the burden processes are on server.

In addition to security issue, performance criteria is another important point of the proposed system. OPES is used not only for making the system more secure but also using the importance of order of the data for Skyline querying. By this encryption schema, the server does not need the decryption process at the querying phase. In addition to this benefit, by using OPES, we get sorted data. Hence, using SFS algorithm makes the system operate more effectively. There is no need of presorting

step for SFS Skyline.

As a result, the Skyline Querying can be securely outsourced by using OPES and meta data hiding. The performance issue is also thought by using SFS and using the checking regions step at the client side, by which not all of the data needed for skyline.

## APPENDIX A: ALGORITHMS

### A.1. Skyline Algorithms

#### A.1.1. BNL

Block Nested Loop, BNL, algorithm is a straightforward computation process for skyline query, in which every point  $p$  is compared with every other point. If the point that is compared with others is not dominated by any of the other points then this point is a skyline point. This algorithm keeps the candidate skyline points in a window in main memory and then fix the real skyline objects by comparing objects in the window with all the objects not in the window. Finally all the objects which can not be dominated by any other objects will be left in the window as the skyline objects [1], [2].

We can summarize the algorithm as follows: The first point  $p$  is inserted to the window. Then we continue with the other points and the subsequent point has three case. In the first case, the subsequent point is dominated by a point from the window, in which case the point is discarded. In the second case, the subsequent point is dominate a point from the window. In this situation, the point is inserted to the window and the dominated points by the subsequent point is dropped from the window. In the last case, the point neither dominates any point from the window nor is dominated by any point. This case needs only the insertion of the subsequent point to the window [1], [2].

The problem of BNL algorithm is the size of the window. The list of skyline points may become large, which makes window not to fit into the memory. In this case, a temporary file is used to store the skyline points that can not fit into the memory. This makes the algorithm to pass many times for this temporary file to guarantee the points in that file [1], [2].

The advantage of this algorithm is that it does not depend to dimensionality of

the data, and as a result no need for sorting or indexing [1], [2].

### **A.1.2. SFS**

Sort Filter Skyline, SFS, requires presorted data on all dimensions of the data table for each attribution according to the descend or ascend sequence. By this pre-sortion step, the need to look for the points that are come after that point. The SFS algorithm also uses main memory, which is named window, as in BNL [2], [22], [23].

All data is sorted in the preprocessing step for each attribute and then the data is ranked according to each of the attributes. The skyline points are the ones that have higher ranks compared to the others. The algorithm scan the presorted data row by row to get the objects with higher ranks in each columns as the skyline objects [2], [22], [23].

### **A.1.3. DC**

Divide and Conquer, DC, algorithm is based on the partitioning approach. The dataset is partitioned into several small partitions by which all partitions could be fit into the main memory [1], [2].

In this approach, skyline is computed for all partitions separately and these steps could be done in main memory. The final skyline points is computed by merging these partial skyline points as a new data and getting skyline points of the data [1], [2].

The DC methodology has some disadvantages. The partitioning step needs all the data to be read initially, and additional IO cost. Also, the post processing in which partial skyline points merged needs another processing effort [1], [2]. However, the small partitions makes the partially computation of skyline points easier.

#### **A.1.4. Epsilon Skyline**

Epsilon skyline is a new approach for skyline querying, which is based on the fixed number of skyline query, or k-skyline that will find exactly k objects in the data table to be skyline points. Controlling the number of skyline points is not possible by common skyline querying algorithms however by this approach user defined skyline computation becomes feasible. This algorithm is not commonly used.

#### **A.1.5. Bitmap-based Skyline**

This approach uses bitmap encoding to decide whether a point is a skyline point or not. All information of a point is encoded in bitmaps. The skyline points are decided according to these bitmaps and operations on these bitmaps [1], [2].

The efficiency of this algorithm comes from the bitwise operations. The bitwise operations speed up the comparison processes. In spite of its effectiveness by bitwise operations, it has some disadvantages like space consumption if the distinct values are large, of not adaptive for user preference queries [1], [2].

#### **A.1.6. Index-based Skyline**

This approach uses d different lists for d-dimensional data. A data point is assigned to a list if the value for that dimension is the minimum value for all dimensions of that value. Then the lists are sorted in ascending order and indexed by B-Tree. The first points in the lists returned as the skyline points in that order [1], [2].

The skyline points which are at the top of the lists can be returned quickly. However, user defined skyline is not supported by this algorithm, the lists are computed for d dimensions and can not be used for n dimensions, which is less than d dimensions [1], [2].

## REFERENCES

1. Papadias, D., Y. Tao, G. Fu, and B. Seeger, “An Optimal and Progressive Algorithm for Skyline Queries”, *ACM SIGMOD*, San Diego, California, USA, 2003.
2. Papadias, D., Y. Tao, G. Fu, and B. Seeger, “Progressive Skyline Computation in Database Systems”, *ACM SIGMOD*, Vol. 30, pp. 41–82, San Diego, California, USA, March 2005.
3. Agrawal, R., J. Kiernan, R. Srikant, and Y. Xu, “Hippocratic Databases”, *Proceedings of the 28th VLDB Conference*, Hong Kong, China, 2002.
4. Agrawal, R., J. Kiernan, R. Srikant, and Y. Xu, “Order Preserving Encryption for Numeric Data”, *ACM SIGMOD*, Paris, France, June 2004.
5. Xia, T. and D. Zhang, “Refreshing the Sky: The Compressed Skycube with Efficient Support for Frequent Updates”, *ACM SIGMOD*, pp. 491–502, Chicago, Illinois, USA, June 2006.
6. Dellis, E., A. Vlachou, I. Vladimirskiy, B. Seeger, and Y. Theodoridis, “Constrained Subspace Skyline Computation”, *CIKM*, pp. 415–424, Arlington, VA, USA, November 2006.
7. Pei, J., Y. Yuan, X. Lin, W. Jin, M. Ester, Q. Liu, W. Wang, Y. Tao, J. X. Yu, and Q. Zhang, “Towards Multidimensional Subspace Skyline Analysis”, *ACM Transactions on Database Systems*, Vol. 31, pp. 1335–1381, December 2006.
8. Pei, J., W. Jin, and Y. Ester, M. Tao, “Catching the Best Views of Skyline: A Semantic Approach Based on Decisive Subspaces”, *Proceedings of the 31st VLDB Conference*, pp. 253–264, Trondheim, Norway, 2005.
9. Yuan, Y., X. Lin, Q. Liu, W. Wang, J. X. Yu, and Q. Zhang, “Efficient Computa-

- tion of the Skyline Cube”, *Proceedings of the 31st VLDB Conference*, pp. 241–252, Trondheim, Norway, 2005.
10. Li, H., Q. Tan, and W. C. Lee, “Efficient Progressive Processing of Skyline Queries in Peer-to-Peer Systems”, *Infoscale*, Hong Kong, June 2006.
  11. Huang, Z. and W. Wang, “A Novel Incremental Maintenance Algorithm of Sky-cube”, *DEXA*, pp. 781–790, Berlin Heidelberg, 2006.
  12. Li, C., B. C. Ooi, A. K. H. Tung, and S. Wang, “DADA: A Data Cube for Dominant Relationship Analysis”, *ACM SIGMOD*, pp. 659–670, Chicago, USA, June 2006.
  13. Pei, J., A. W. C. Fu, X. Lin, and H. Wang, “Computing Compressed Multidimensional Skyline Cubes Efficiently”, .
  14. Tian, L., L. Wang, P. Zou, Y. Jia, and A. Li, “Continuous Monitoring of Skyline Query over Highly Dynamic Moving Objects”, *MobiDE*, pp. 59–66, Beijing, China, June 2007.
  15. Gao, Y., G. Chen, L. Chen, and C. Chen, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data”, *BNCOD*, pp. 127–139, 2006.
  16. Wu, P., C. Zhang, B. Y. Zhao, D. Agrawal, and A. E. Abbadi, “Parallelizing Skyline Queries for Scalable Distribution”, *EDBT*, pp. 112–130, 2006.
  17. Sharifzadeh, M. and C. Shahabi, “The Spatial Skyline Queries”, *VLDB*, pp. 751–762, Seoul, Korea, September 2006.
  18. Chan, C. Y., H. V. Jagadish, K. L. Tan, A. K. H. Tung, and Z. Zhang, “Finding k-Dominant Skylines in High Dimensional Space”, *ACM SIGMOD*, pp. 503–514, Chicago, Illinois, USA, June 2006.
  19. Yang, Z., B. Wang, and M. Kitsuregawa, “General Dominant Relationship Analysis based on Partial Order Models”, *SAC*, pp. 470–474, Seoul, Korea, March 2007.



20. Chan, C. Y., P. K. Eng, and K. L. Tan, “Stratified Computation of Skylines with Partially-Ordered Domains”, *ACM SIGMOD*, pp. 203–214, Baltimore, Maryland, USA, June 2005.
21. Borzsonyi, S., D. Kossman, and K. Stocker, “The Skyline Operator”, *ICDE*, pp. 421–430, 2001.
22. Godfrey, P., R. Shipley, and J. Gryz, “Algorithms and Analyses for Maximal Vector Computation”, *Springer-Verlag*, 2006.
23. Chomicki, J., P. Godfrey, J. Gryz, and D. Liang, “Skyline with Presorting: Theory and Optimizations”, .
24. Stolba, N. and A. M. Tjoa, “An Approach towards the Fulfilment of Security Requirements for Decision Support Systems in the Field of Evidence-Based Healthcare”, .
25. Stolba, N., M. Banek, and A. M. Tjoa, “The Security Issue of Federated Data Warehouses in the Area of Evidence-Based Medicine”, .
26. Thomas, D., “Research Statement: Privacy in Databases”, December 2006.
27. Goyal, V., A. Sahai, O. Pandey, and B. Waters, “Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data”, *CCS*, pp. 89–98, Virginia, USA, October 2006.
28. Yang, Y., N. Wilfred, H. L. Lau, and J. Cheng, “An Efficient Approach to Support Querying Secure Outsourced XML Information”, *CAISE*, pp. 155–171, 2006.
29. Imamura, T., A. Clark, and H. Maruyama, “A Stream-Based Implementation of XML Encryption”, *ACM Workshop on XML Security*, pp. 11–17, VA, USA, November 2002.
30. Bender, W., W. Butera, D. Gruhl, R. Hwang, F. J. Paiz, and S. Fogreb, “Appli-

cation for Data Hiding”, *IBM Systems Journal*, pp. 547–568, 2000.