Private Cell Retrieval from Data Warehouse

Journal:	*Transactions on Information Forensics & Security*
Manuscript ID:	T-IFS-05412-2015
Manuscript Type:	Regular Paper
Date Submitted by the Author:	05-Jul-2015
Complete List of Authors:	Yi, Xun; RMIT University, Computer Science and IT Paulet, Russell; Victoria University, Engineering and Science Bertino, Elisa; Purdue University, Computer Science Xu, Guandong; University of Technology Sydney, Advanced Analytics Institute
EDICS:	ADP-PPRO-Privacy protection < ADP-ANONYMIZATION AND DATA PRIVACY, ADP-PATT-Privacy attacks < ADP-ANONYMIZATION AND DATA PRIVACY, ADP-PMOD-Privacy modeling and analysis < ADP-ANONYMIZATION AND DATA PRIVACY

Private Cell Retrieval from Data Warehouse

Xun Yi, Russell Paulet, Elisa Bertino, *Fellow, IEEE* and Guangdong Xu

Abstract—Private Information Retrieval (PIR) techniques allow the client to retrieve a cell from a data warehouse without revealing to the operator which cell is retrieved. However, PIR cannot be used to hide OLAP operations performed by the client, which may disclose the client's interest. This paper presents a solution for private cell retrieval from data warehouse on the basis of the Paillier public key cryptosystem. By our solution, the client can privately perform OLAP operations on the data warehouse and retrieve one (or more) cell without revealing any information about which cell is selected. In addition, we propose a solution for private block download on the basis of the Paillier cryptosystem. Our private block download allows the client to download an encrypted block from a data warehouse without revealing which block is downloaded and improves the feasibility of our private cell retrieval. Our solutions ensure both server's privacy and client's privacy.

Index Terms—Private information retrieval, data warehouse, Paillier cryptosystem and Boneh-Goh-Nissim cryptosystem

1 Introduction

Data warehousing provides tools for business executives to systematically organise, understand, and use their data to make strategic decisions. A large number of organizations have found that data warehouses are valuable in today's competitive, fast-evolving world. In the last several years, many firms have spent millions of dollars in building enterprise-wide data warehouse. Many people feel that with competition mounting in every industry, data warehouse is the latest must-have marketing weapon - a way to keep customers by learning more about their needs [20].

A data warehouse is a **subject-oriented**, **integrated**, **time - variant** and **non-volatile** collection of data in support of management's decision making process [21]. Data warehouses are built on a multidimensional data model. This model views data in the form of a data cube. A data cube, defined by dimensions and measures, allows data to be viewed in multiple dimensions. In general, dimensions are the entities with respect to which we want to keep records. For example, a sales data warehouse may keep records of the store's sales with respect to dimensions - time, location and product. Measures are the quantities by which we want to analyse relationships between dimensions. Examples of measures for a sales data warehouse include the sales amount, the number of units sold, and the average sales amount. An example is the sales data organized with respect to 3 dimensions - time, products and locations.

In the multidimensional model, data is organised into multiple dimensions, where each dimension has multiple-levels of abstraction defined by a concept hierarchy. A concept hierarchy defines a sequence of mappings from a set of low-level concepts to high-level, more general concepts. The concept hierarchy for locations could be street, city, state, and country. This organisation provides clients with the flexibility to view data from different perspectives. A number of online analytical processing (OLAP) operations exist to materialise these different views, supporting interactive querying and analysis of the data at hand. Typical OLAP operations include: **roll-up** - aggregation by climbing up a concept hierarchy; **drill-down** - the reverse of roll-up; **slice** - a selection on one dimension, resulting in a sub-cube; **dice** - a selection on two or more dimensions, resulting in a sub-cube; **pivot** - rotating the data axes in view in order to provide an alternative presentation of the data.

Queries to the data warehouse are based on a star-net model, which consists of radial lines emanating from a central point, where each line represents a concept hierarchy of a dimension. These represent the granularities available for use by OLAP operations such as drill-down and roll-up. A star-net query model for a sales data warehouse is shown in Fig. 1, where each circle is called a footprint.

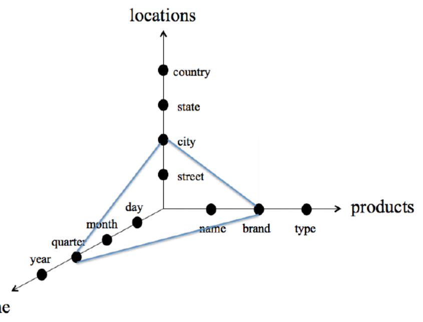

Fig. 1. A star-net model for data warehouse queries

Data warehousing is often built on the client / server model, where a data warehouse server provides data query service to clients. A client may perform certain OLAP operations on the data warehouse with the help of the server before retrieving one or more cells (measures) of the data warehouse from the server. Examples of data warehouse in the client/server model are a stock exchange data warehouse and a pharmaceutical data warehouse.

There are two privacy issues with data warehouse queries.

- X. Yi is with the School of computer Science and Information Technology, RMIT University, VIC 3001, Australia.
  E-mail: xun.yi@rmit.edu.au
- R. Paulet is with the College of Engineering and Science, Victoria University, Melbourne, VIC 8001, Australia. E-mail: russell.paulet@vu.edu.au
- E. Bertino is with the Computer Science Department, Purdue University, West Lafayette, IN 47907. E-mail: bertino@purdue.edu
- G. Xu is with the Advanced Analytics Institute at University of Technology Sydney, Sydney, NSW 2007, Australia. E-mail: guandong.xu@uts.edu.au

Manuscript received July 6, 2015.

One privacy issue is the client's privacy. In order to query the data warehouse, a user usually requests the server to perform OLAP operations and send back a cell. An important issue in this simple process is represented by the privacy of the user query as the user query may reveal to the server business sensitive information. For example, for a stock exchange data warehouse, the user may be an investor, who queries the data warehouse for the trend of a certain stock. He may wish to keep private the identity of the stock he is interested in. For a pharmaceutical data warehouse, the user may be a laboratory, which would like to keep private the active principles it wants to use. To protect his privacy, the user accessing a data warehouse may therefore want to perform OLAP operations and retrieve a cell without revealing any information about which cell he is interested in.

A trivial solution to the above private data warehouse query problem is for the user to download the entire data warehouse and then locally perform OLAP operations and retrieve the cell of interest. This solution is not suitable if the owner of the data warehouse wishes to make profit through data warehouse services (for example, a health care data warehouse). Usually, the user is interested in only a part of the data warehouse. Purchasing the entire data warehouse may not be an economically viable.

Private Information Retrieval (PIR) protocols, such as [11], [25], do not fully address the private data warehouse query problem. A PIR protocol allows a user to retrieve a record from a database without the owner of that database being able to determine which record was selected with communication cost less than the database size. By using PIR, a user can retrieve a cell (a record) from a data warehouse (a database) without revealing any information about which cell is retrieved. However, the user cannot hide his OLAP operations to the server when he requests the server to perform the operations. These operations may reveal the user's interest. For example, when the user requests the server to perform a slice operation with respect to a location, the server can learn the user's interest in the location. It is a challenge to assure the user's privacy when performing OLAP operations.

Another privacy issue with data warehouse queries is the server's privacy. Usually, data warehouse is built for certain business purposes and the owner of data warehouse (server) wishes to make profit by offering data warehouse query services, such as one query per pay. The server has to disclose some data to the client when the client queries the data warehouse, but the server wants to keep the rest of the data private. The service's privacy was called the server's security in [41]. Security of a databased server usually refers to securing network against hackers. To avoid confusion, it is better to use the server's privacy term.

In our previous work [41], we gave a solution for private data warehouse queries on the basis of Boneh-Goh-Nissim (BGN) [7] cryptosystem. Our basic idea is to allow the data warehouse owner to encrypt its data warehouse and distribute the encrypted data warehouse to the user who wishes to perform private data warehouse queries. The user can perform any OLAP operations on the encrypted data warehouse locally without revealing his interest. When the user wishes to decrypt a cell of the encrypted data warehouse, the user and the server run a Private Cell Retrieval (PCR) protocol jointly to decrypt the cell without revealing to the server which cell is retrieved.

Unlike operational databases, a data warehouse is non-volatile. The data in the data warehouse is never over-written or deleted - once committed; the data is static, read-only, and retained for future reporting. It is feasible to allow the data warehouse owner to distribute the encrypted data warehouse to potential users only once and let the users download new added data online if any.

Assume that the server charges the client per query, our solution based on the BGN cryptosystem [41] allows the user to perform some statistical analysis, such as regression and variance analysis, on the encrypted data warehouse with the lowest cost. The reason is that the BGN cryptosystem allows one to evaluate multi-variate polynomials of total degree 2 on encrypted values. However, it needs relatively long time to decrypt a ciphertext, which is an encryption of a large plaintext.

To overcome this problem, we give a solution for private cell retrieval from data warehouse on the basis of the Paillier public key cryptosystem [35] in this paper. This solution is faster than our solution based on BGN cryptosystem.

To enhance our solutions for private data warehouse queries, we also give an approach for private block download (PBD) on the basis of the Paillier cryptosystem [35]. Our PBD protocol allows a client to download a block from the encrypted data warehouse without revealing which block is downloaded. If the client cannot get the entire encrypted data warehouse and he is interested in only one block of the data warehouse, he can download the block from the server with our PBD protocol and then retrieve cells from the block with our PCR protocol. In this way, our PBD protocol can improve the feasibility of our private data warehouse queries. Our PBD protocol can be used to perform dice or slice operation on the encrypted data warehouse in the server side without revealing to the server which block of the data warehouse is retrieved.

Our solutions ensures both the server's privacy in the sense that the server, for billing purpose, releases to the user data paid by the user, and the client's privacy in the sense that the client does not reveal any information about his queries to the server. We have implemented our solution on an example of data warehouse and experiments have shown that our solution is practical for private data warehouse queries.

This paper extends our previous paper [41] by including a new PCR protocol based on the Paillier cryptosystem to improve the performance and a new PBD protocol to reduce the size of the encrypted data warehouse that the client has to download. This is a significant extension that enhances the flexibility and efficiency of our approach. In this paper we also analyze the security and complexity of such protocols. A combination of our PBD and PCR protocol has comparable performance against existing PIR protocols. Furthermore, we model multiple queries with both our PCR and PBD protocols and analyze the client's privacy with multiple queries.

Our solutions for private data warehouse queries can be adapted to the cloud computing environment, where the data warehouse owner outsources the encrypted data cube to the cloud and the outsourced cloud server provides the private block download service to the client. The cloud server does not know the decryption key of the data warehouse owner and thus the privacy of the data warehouse can be preserved. The data warehouse owner is needed only when the client wants to decrypt a ciphertext through our private cell retrieval. As shown in our performance analysis, our private cell retrieval can be done efficiently.

The rest of the paper is organized as follows. Related work is surveyed in Section 2. We define our model and described our solution from Sections 3 to 4. The security and performance analysis is carried out in Sections 5 and 6. Experimental results are shown in Section 7. Conclusions are drawn in the last section.

2 Related Work

A closely related work is Private Information Retrieval (PIR), which was firstly introduced by Chor, Goldreich, Kushilevitz, and Sudan in 1995 [11]. In their paper, they proposed a set of schemes to implement PIR through replicated databases, which provide users with information - theoretic security as long as some of the database replicas do not collude against the user. Since then, a lot of research on PIR has been done. We classify the results as follows [3], [34].

Information-Theoretic Private Information Retrieval

"Information-theoretic" stands for the fact that the user privacy is assumed to be unbreakable independently from the computational power of a cheater. Chor et al. proved, that any information-theoretic PIR solution has a communication cost with a lower bound equal to the database size [11]. Then they relaxed the problem setting and assumed that there are several (instead of one) database servers, which do not communicate among each other, storing with the same data. This assumption makes the non-trivial information-theoretic PIR feasible. The basic idea is to send several queries to several databases. The queries are constructed in such a way, that they give no information to the servers about the record that the user is interested in. But using the answers from the queries, the user can construct the desired record. Chor et al. also considered the case when up to t of the servers are allowed to collude against the user.

Ambainis [1] improved the results of Chor et al. [11], which led to two non-trivial information-theoretic PIR solutions: (1) a k database scheme (i.e., a scheme with k identical databases non-communicating to each other), for any constant $k \geq 2$, with communication complexity $O(N^{1/(2k-1)})$; and (2) a $\Theta(\log N)$ database scheme with communication complexity $O(\log^2 N \cdot \log\log N)$, where N is the size of the database. Further research on information-theoretic PIR appeared in [5], [22], [22], [23], [24].

Computational Private Information Retrieval

In order to get better communication complexity, the computational assumption was weakened by Chor and Gilboa in [10]. "Computational" means that the database servers are presumed to be computationally bounded, i.e., under an appropriate intractability assumption, the database cannot gain information about which data element was selected by the user. For every $\epsilon > 0$, Chor and Gilboa presented two computational PIR schemes with complexity $O(N^\epsilon)$.

In the first paper on PIR [11] it was proven that the information-theoretic PIR problem has no non-trivial solutions for the case of a single database. Surprisingly, the substitution of an information-theoretic security with an intractability assumption achieves a non-trivial PIR protocol for single database scheme [25]. Its communication complexity is $O(N^\epsilon)$ for any $\epsilon > 0$. They use an intractability assumption described in [19]. The basic approach is to encrypt a query in such a way that the server still can process it using special algorithms. However, the server recognizes neither the clear-text query nor the result. The result can be decrypted only by the user. This was the first single-database protocol that considers database privacy. Using another intractability assumption, Cachin et al. [8] demonstrated a single database computational PIR protocol that has poly-logarithmic communication. This is an improvement compared with the polynomial communication complexity in [25]. This result looks particularly effective, because the user has to send a minimum $\log N$ bits just to address the bit he wants to retrieve in the database. Protocols with better results appeared in [9], [15], [27], [28]. Recently, Yi et al. [40] proposed a single-database PIR with computational efficiency on the basis of the state of the art fully homomorphic encryption technique [13], [16], [38].

Symmetrical Private Information Retrieval

Symmetrical PIR is a PIR problem, where the privacy of the database is considered, i.e., a symmetrical PIR protocol must prevent the user from learning more than one database record of the database during a session. Clearly, symmetrical privacy (database privacy) is a very important property for practical applications, since an efficient billing is only then possible. A symmetrical PIR protocol for single server was first proposed in [26], and for several servers it was considered in [17]. Other symmetrical PIR were proposed in [30], [31], [32].

In addition, Private Block Retrieval (PBR) is a natural extension of PIR in which, instead of retrieving only a single bit, the user retrieves a d-bit block that begins at an index i. PBR techniques are important for making PIR practical. Information-theoretic PBR was introduced in [10]. A practical PBR protocol for a single database was given by Gentry and Ramzan [15]. The security of this scheme is based on a simple variant of the Φ-hiding number-theoretic assumption by Cachin, Micali and Stadler [8]. This scheme has communication complexity $O(k + d)$ only, where $k \geq \log N$ is a security parameter that depends on the database size N and d is the bit-length of the retrieved database block.

With PIR or PBR, a user can retrieve a cell from a data warehouse without revealing any information about which cell is retrieved. However, the user cannot hide his OLAP operations to the server when he requests the server to perform the operations. These operations may reveal the user's interest. In fact, the server can predict the most likely next queries by formulating OLAP queries of the user [4].

3 Models for Private Data Warehouse Queries

In this section, we construct two models for private data warehouse queries. One model is for private cell retrieval and another model is for private block download.

3.1 Model for Private Cell Retrieval

In this model, we consider a data cube D with d dimensions $y_1, y_2, \cdots y_d$ and t measures $x_1, x_2, \cdots, x_t$, denoted as

$$D = (x_1, x_2, \cdots, x_t)_{y_1, y_2, \cdots, y_d},$$

where the data cube dimension domain $DD = DD_1 \times DD_2 \times \cdots DD_d$ and DD_i is the domain of dimension y_i, $1 \leq i \leq d$.

We assume that the data cube is managed by a server $\mathcal{S}$ and used by clients. The server $\mathcal{S}$ wishes to make a profit by providing data warehouse services to clients. The clients wish to learn some knowledge from D through OLAP operations on D without revealing their interests to $\mathcal{S}$.

First of all, on input of a security parameter κ, the server $\mathcal{S}$ generates its public/private key pair $\{PK, SK\}$ and encrypts the data cube D into $E(D)$ with the public key PK, where the values of all measure attributes are encrypted, but the values of all dimension attributes are in plaintexts. The encrypted data cube $E(D)$ can be then released to clients.

A client $\mathcal{C}$ can either download the encrypted data cube $E(D)$ from the server's web site or request the server to send a CD (Compact Disc) of the encrypted data cube by post. This download (transmission) is executed only once because the data warehouse is non-volatile. For new data added into the data cube, we allow the users to download it online. The client can then perform any OLAP operation on the encrypted data cube $E(D)$ locally.

In many cases, the "client" is actually an organization that then has no problem in downloading and storing the encrypted data warehouse $E(D)$ and performing OLAP operations on $E(D)$.

In order to retrieve a cell from the data cube D after several OLAP operations on $E(D)$ (i.e., to decrypt a ciphertext C from the encrypted data cube $E(D)$ after several OLAP operations on $E(D)$), the server $\mathcal{S}$ and the client $\mathcal{C}$ runs a Private Cell Retrieval (PCR) protocol, composed of three algorithms as follows.

(1) Query Generation (QG): Takes as input the public key PK of the server $\mathcal{S}$, the ciphertext C, which is an encryption of either a measure value or a function of several measure values, (the client) outputs a query Q and a secret s, denoted as $(Q, s) = \mathsf{QG}(C, PK)$.

(2) Response Generation (RG): Takes as input the query Q and the private key SK of the server $\mathcal{S}$, (the server) outputs a response R, denoted as $R = \mathsf{RG}(Q, SK)$.

(3) Response Retrieval (RR): Takes as input the public key PK of the server $\mathcal{S}$, the response R and the secret s of the client, (the client) outputs a plaintext m, denoted as $m = \mathsf{RR}(R, PK, s)$.

A PCR protocol can be illustrated as in Fig. 2 and is correct if, for any security parameter κ, for any ciphertext C,

$$Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$$

holds, where $(Q, s) = \mathsf{QG}(C, PK)$, $R = \mathsf{RG}(Q, SK)$ and $Decrypt$ denotes the decryption algorithm for the server.

An execution of the PCR protocol is actually a decryption query. The server $\mathcal{S}$ can charge the client $\mathcal{C}$ per decryption query. The cost for the client to query the data warehouse is linear in the number of decryption queries or the number of the executions of the PCR protocol.

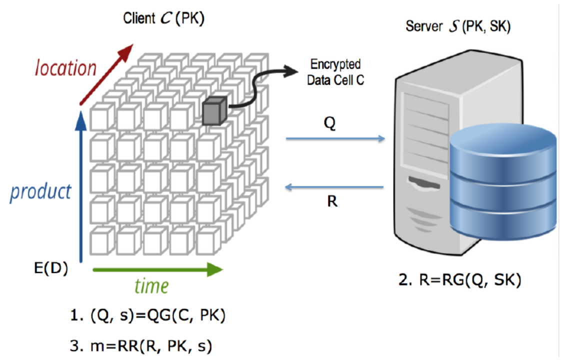

Fig. 2. Private Cell Retrieval

3.2 Model for Private Block Download

To supplement to our first model where the client is provided with the entire encrypted data warehouse before private queries, we give another model for a client to download a block (sub-cube) from the encrypted data warehouse without revealing to the server which block (sub-cube) is downloaded. This process can be considered as a private slice or dice operation over the encrypted data warehouse at the server side. The purpose of private block download is to avoid downloading the entire data warehouse when the client is interested in only one part of the data warehouse.

The model for private block download is built on the concept of k-anonymity [39] (the information for each person contained in the release cannot be distinguished from at least k-1 individuals whose information also appear in the release). In addition, we borrow the concept of cloaking region from private location-based queries [18] because our model also deals with private queries. The cloaking region CR is chosen to hide the privacy of the client's interest.

To download a block B from $E(D)$, a client $\mathcal{C}$ chooses a cloaking region CR within $E(D)$ at first. The CR may be the entire $E(D)$ or a part of $E(D)$. The cloaking region CR is composed of n blocks, one of them is the block B and the rest are randomly chosen from $E(D)$ by the client. The n blocks are denoted as $B_1, B_2, \cdots, B_n$, where $B_i = B$, $1 \leq i \leq n$. For example, the client $\mathcal{C}$, who is interested in the data in Los Angeles, may choose Los Angeles, San Diego, San Jose, and San Francisco as the cloaking region CR to achieve 4-anonymity.

We assume that the client $\mathcal{C}$ randomly chooses $n-1$ blocks with the same dimensions and the same granularities as the block B. For example, if the block B is the data for Los Angeles city with two dimensions - years and brands, the client may choose three blocks for San Diego, San Jose, and San Francisco with the same dimensions. The cloaking region is composed of the four blocks. Then the client and the server run a Private Block Download (PBD) protocol, composed of three algorithms as follows.

(1) Query Generation (QG): Takes as input the index i of the block B_i, the cloaking region CR, the total number n of blocks, the public key PK of the server and the security parameter κ, (the client) generates a public and private key pair (pk, sk) and outputs a query Q, denoted as $(Q, sk) = \mathsf{QG}(i, CR, n, PK)$, where Q contains the public key pk of the client, CR and n in plaintext.

(2) Response Generation (RG): Takes as input the query Q (including the public key pk of the client, and CR and n), (the server) extracts n blocks according to CR and outputs a response R, denoted as $R = \mathsf{RG}(Q, PK)$.

(3) Response Retrieval (RR): Takes as input the response R and the private key sk of the client $\mathcal{C}$, (the client) outputs a block B_i', denoted as $B_i' = \mathsf{RR}(R, sk)$.

A PBD protocol can be illustrated as in Fig. 3 and is correct if, for any security parameter κ, for any i, CR and n, we have $B_i = \mathsf{RR}(R, sk)$, where $(Q, sk) = \mathsf{QG}(i, CR, n, PK)$ and $R = \mathsf{RG}(Q, PK)$.

Because the execution of the PBD protocol consumes computational resources, the server $\mathcal{S}$ can charge the client C according to CR and n. The computational complexity of PBD for the server is linear in the size of CR and n, and the communication overhead for the client to download a block is linear in the size of CR/n.

To privately retrieve a cell from the data warehouse, the client $\mathcal{C}$ may run a PBD protocol with the server $\mathcal{S}$ to download an encrypted block B_i which contains the encrypted cell from the encrypted data warehouse $E(D)$ at first and then run a PCR protocol with the server $\mathcal{S}$ to decrypt the cell.

The differences between the above two models are as follows:

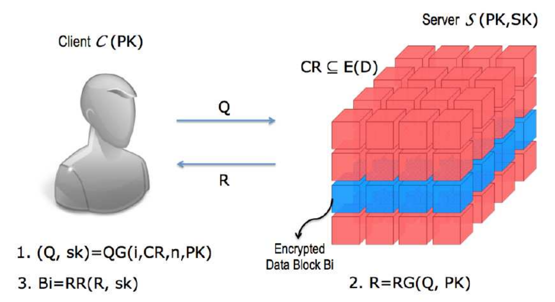

Fig. 3. Private Block Download

- In PCR, the client $\mathcal{C}$ has been provided the encrypted data warehouse $E(D)$ before private queries. In PBD, the client $\mathcal{C}$ does not have $E(D)$ and wishes to download an encrypted block from $E(D)$ without revealing to the server $\mathcal{S}$ which block is downloaded.
- In PCR, the server $\mathcal{S}$ generates its response R with its private key SK. In PBD, the server $\mathcal{S}$ generates R with the public key pk of the client and therefore the role of the data warehouse server can be played by any server which hosts the encryption of data warehouse, such as a cloud service provider.

3.3 Privacy Definitions

The privacy of the PCR/PBD protocols involves the server's privacy and the client's privacy. Intuitively, the server $\mathcal{S}$ wishes to release only one measure value to the client $\mathcal{C}$ each time the client sends a query. Meanwhile, the client $\mathcal{C}$ does not wish to reveal to the server which cell or block is retrieved.

Formally, the server's privacy can be defined with a game as follows.

Given a data cube D and the public key PK of the server, consider the following game between an adversary (the client) $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) Given the public key PK of the server, the adversary $\mathcal{A}$ chooses two different values m_1, m_2 of either two different measure attributes or the same measure attribute and sends them to $\mathcal{C}$.
(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, and encrypts m_b to obtain $C_b = Encrypt(m_b, PK)$, and then sends C_b back to $\mathcal{A}$.
(3) The adversary $\mathcal{A}$ can experiment with the code of C_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0,1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be

$$\mathsf{Adv}_{\mathcal{A}}(k) = |\Pr(b' = b) - 1/2|,$$

where k is the security parameter.

Definition 1 (Server's Privacy Definition) In the PCR/PBD protocols, the data warehouse server has (semantic) data privacy if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ in the above game is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

Server's privacy ensures that the client cannot decrypt any ciphertext in the encrypted data cube $E(D)$ without the help of the server. In PBD, the client $\mathcal{C}$ downloads an encrypted block without need of any decryption. Therefore, it is obvious that the PBD has the server's privacy.

Next, we formally define the client's privacy for PCR and PBD protocols in a single query with a game as follows.

Give an encrypted data cube $E(D)$ and the public/private key pair (PK, SK) of the server, as well as the cloaking region CR and the number n of blocks in the case of PBD, consider the following game between an adversary (the server) $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) The adversary $\mathcal{A}$ chooses two different ciphertexts C_0 and C_1 (in the case of PCR) or two different blocks B_0 and B_1 of n blocks with the same dimensions within a cloaking region CR (in the case of PBR), and then sends them to the challenger $\mathcal{C}$.
(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, and executes the Query Generation (QG) to obtain $(Q_b, s_b) = \mathsf{QG}(C_b, PK)$ (in the case of PCR) or $(Q_b, sk_b) = \mathsf{QG}(b, CR, n, PK)$ (in the case of PBR), where s_b and sk_b are the secrets of the challenger $\mathcal{C}$, and then sends Q_b back to the adversary $\mathcal{A}$.
(3) The adversary $\mathcal{A}$ can experiment with the code of Q_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0,1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be

$$\mathsf{Adv}_{\mathcal{A}}(k) = |\Pr(b' = b) - 1/2|,$$

where k is the security parameter.

Definition 2 (Client's Privacy Definition for Single Query) In the PCR or PBR protocol, the client has (semantic) query privacy if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ in the above game is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

Client's privacy ensures that the server cannot tell which information the client has retrieved from the data cube $E(D)$ in PCR protocol and which block in CR the client wants to download in PBD protocol.

Like k-anonymity, the client's privacy in PBD protocol depends on the number n of blocks in the cloaking region CR. The larger n is, the higher privacy level can be achieved. The maximum of CR is the entire encrypted data warehouse $E(D)$.

At last, we formally define the client's privacy for PCR and PBD protocols in multiple queries with a game as follows.

Give an encrypted data cube $E(D)$ and the public/private key pair (PK, SK) of the server, consider the following game between an adversary (the server) $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) The adversary $\mathcal{A}$ chooses two different sequences of ciphertexts or blocks of ciphertexts, $A_{0,1}, A_{0,2}, \cdots, A_{0,\ell}$ and $A_{1,1}, A_{1,2}, \cdots, A_{1,\ell}$, where $A_{i,j}$ is either a ciphertext or a block of ciphertexts. When $A_{i,j}$ is a block of ciphertexts, $A_{0,j}$ and $A_{1,j}$ are two of n_j blocks of ciphertexts and all blocks have the same dimensions and the same granularities. Then $\mathcal{A}$ sends them to the challenger $\mathcal{C}$.
(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, and executes the Query Generation (QG) on $A_{b,j}$ to

obtain $(Q_{b,j}, s_{b,j}) = \mathsf{QG}(A_{b,j}, PK)$ (in the case of PCR) or $(Q_{b,j}, sk_{b,j}) = \mathsf{QG}(b, CR_j, n_j, PK)$ (in the case of PBR), where $j = 1, 2, \cdots, \ell$ and $s_{b,j}$ and $sk_{b,j}$ are the secrets of the challenger $\mathcal{C}$, and then sends $Q_{b,1}, Q_{b,2}, \cdots, Q_{b,\ell}$ back to the adversary $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of $Q_{b,1}, Q_{b,2}, \cdots, Q_{b,\ell}$ in an arbitrary non-black-box way, and finally outputs $b' \in \{0, 1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be $\mathsf{Adv}_{\mathcal{A}}(k) = |\Pr(b' = b) - 1/2|$, where k is the security parameter.

Definition 3 (Client's Privacy Definition for Multiple Queries) In multiple queries with PCR/PBD protocols, the client has (semantic) query privacy if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ in the above game is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

4 Private Data Warehouse Queries

In this section, we construct a private cell retrieval (PCR) protocol according to our PCR model and a private block download (PBD) protocol according to our PBD model. Then we describe how to perform private OLAP operations over the encrypted data warehouse.

4.1 Paillier-Based Private Cell Retrieval Protocol

Based on our PCR model, we gave a construction of PCR protocol based on the BNG cryptosystem [7] in [41], which allows the client to retrieve a measure value in a cell without revealing the measure and cell attributes to the server. Now we give a new construction of PCR protocol based on the Paillier cryptosystem [35]. We consider a data cube D with d dimensions $y_1, y_2, \cdots y_d$ and t measures $x_1, x_2, \cdots, x_t$, denoted as

$$D = (x_1, x_2, \cdots, x_t)_{y_1, y_2, \cdots, y_d}.$$

Our PCR protocol based on the Paillier homomorphic encryption scheme [35] assumes that the server $\mathcal{S}$ randomly chooses two large primes p, q on the basis of a security parameter κ, lets $SK = \{p, q\}$ and $PK = \{g, N\}$, where g is chosen from $\mathbb{Z}_{N^2}$ and its order is a nonzero multiple of N. Before releasing the data cube to clients, the server $\mathcal{S}$ runs the Initialisation algorithm to encrypt the data cube D to $E(D)$, as described in Algorithm 1.

Algorithm 1 Initialisation (Server)

```
Input: $D = (x_1, x_2, \cdots, x_t)_{y_1, y_2, \cdots, y_d}, PK$
Output: $E(D) = (E(x_1), E(x_2), \cdots, E(x_t))_{y_1, y_2, \cdots, y_d}$
1: Let $E(D) = D$
2: For each measure value $x = (x_i)_{y_1, y_2, \cdots, y_d}$ in $E(D)$, where
   $1 \le i \le t$ and $(y_1, y_2, \cdots, y_d) \in DD$, where $DD$ is the
   dimension domain.
3: { Pick a random integer $r$ from $\{1, 2, \cdots, N\}$
4:   Encrypt $x$ by computing
       $z = Encrypt(x, PK) = g^x r^N (mod\ N^2)$
     and replace $(x_i)_{y_1, y_2, \cdots, y_d}$ with $z$, denoted as
     $(E(x_i))_{y_1, y_2, \cdots, y_d}$, where the encryption algorithm is
     based on the Paillier cryptosystem [35].
5: } //End of For
6: return $E(D)$
```

Given the encrypted data cube $E(D)$, if a client $\mathcal{C}$ wishes to retrieve a measure value in a cell, in other words, to decrypt a ciphertext C in a cell, the client $\mathcal{C}$ and the server $\mathcal{S}$ run our Private Cell Retrieval protocol, composed of three algorithms, Query Generation (QG), Response Generation (RG), and Response Retrieval (RR), as in Algorithms 2-4.

Algorithm 2 PCR Query Generation QG (Client)

```
Input: $C, PK$
Output: $Q, s$
1: Pick a random integer $s$ from $\{1, 2, \cdots, N\}$
2: Compute $Q = C^s (mod\ N^2)$
3: return $(Q, s)$
```

Algorithm 3 PCR Response Generation RG (Server)

```
Input: $Q, SK = \{p, q\}$
Output: $R$
1: Let $\lambda = lcm(p-1, q-1)$
2: Compute
     $R = Decrypt(Q, SK) = \frac{(Q^\lambda (mod\ N^2) - 1)/N}{(g^\lambda (mod\ N^2) - 1)/N} (mod\ N),$
   where the decryption algorithm is based on the Paillier cryptosystem [35].
3: return $R$
```

Algorithm 4 PCR Response Retrieval RR (Client)

```
Input: $R, PK, s$
Output: $m$
1: Compute $m = R \cdot s^{-1} (mod\ N)$
2: return $m$
```

Theorem 1 (Correctness) Our Paillier-based PCR protocol is correct. In other words, for any security parameter κ, for any ciphertext C,

$$Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$$

holds, where $(Q, s) = \mathsf{QG}(C, PK)$ and $R = \mathsf{RG}(Q, SK)$.

Proof We assume that $C = g^{m'} r^N (mod\ N^2)$. With reference to [35], we have $Decrypt(C, SK) = m'$. In addition,

$$Q \quad = \quad C^s = (g^{m'} r^N)^s = g^{m's} (r^s)^N (mod\ N^2).$$

Therefore, $R = Decrypt(Q, SK) = m's (mod\ N)$ and we have $\mathsf{RR}(R, PK, s) = R \cdot s^{-1} = m's \cdot s^{-1} = m' (mod\ N)$, i.e., $Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$. The theorem is proved. △

Our Paillier-based PCR protocol does not require the client to compute the discrete logarithm and thus is more efficient than our BGN-based PCR protocol.

4.2 Private Block Download Protocol

Based on our PBD model, we give a construction of the PBD protocol which allows the client to download a block from the encrypted data warehouse without revealing to the server which block is downloaded.

Our PBD protocol is built on the Paillier homomorphic encryption scheme [35]. It is needed only when the encrypted data warehouse is huge and it is hard for the client to download and store the entire encrypted data warehouse.

Before the client and the server run our PBD protocol, the client may request the server to perform certain OLAP operations, such as roll-up or drill-down, without revealing the real query of the client. Suppose that the client wants to download a block B from the encrypted data warehouse $E(D)$. He randomly chooses $n-1$ blocks with the same dimensions as B and the n blocks $B_1, B_2, \cdots, B_n$ (where $B_i = B$ and $1 \leq i \leq n$) form a cloaking region CR.

Without loss of generality, we assume that each block B_i contains m ciphertexts $C_{i1}, C_{i2}, \cdots, C_{im}$ in a order.

Algorithm 5 PBD Query Generation (Client)

```
Input: $i, CR, n, PK$ (the public key of the server $\mathcal{S}$)
Output: $Q, sk$
1: Randomly choose two large primes $p, q$ on the basis of a security parameter $\kappa$ and $PK$, such that the size of $N = pq$ is more than the size of the ciphertext in $E(D)$.
2: Let $sk = \{p, q\}$ and $pk = \{g, N\}$, where $g$ is chosen from $\mathbb{Z}_{N^2}$ and its order is a nonzero multiple of $N$.
3: For each $j \in \{1, 2, \cdots, n\}$
4: { Pick a random integer $r_j \in \mathbb{Z}^*_{N^2}$, compute
       $z_j = \begin{cases} Encrypt(1, pk) = g^1 r_j^N (mod\ N^2) & \text{if } j = i \\ Encrypt(0, pk) = g^0 r_j^N (mod\ N^2) & \text{otherwise} \end{cases}$
   where the encryption algorithm is based on the Paillier cryptosystem [35].
5: } //End of For
6: Let $Q = \{CR, n, z_1, z_2, \cdots, z_n, pk\}$, $sk = \{p, q\}$.
7: return $Q, sk$
```

Algorithm 6 PBD Response Generation RG (Server)

```
Input: $Q = \{CR, n, z_1, z_2, \cdots, z_n, pk = (g, N)\}, PK$
Output: $R = \{C_1, C_2, \cdots, C_m\}$
1: Based on CR and $n$ in $Q$, extract the $n$ blocks $B_1, B_2, \cdots, B_n$ with the same dimensions, where $B_j = \{C_{j1}, C_{j2}, \cdots, C_{jm}\}$ $(j = 1, 2, \cdots, n)$
2: For $j = 1, 2, \cdots, n$, compute
       $S_j = \{z_j^{C_{j1}}(mod\ N^2), z_j^{C_{j2}}(mod\ N^2), \cdots, z_j^{C_{jm}}(mod\ N^2)\}$
3: Compute $R = \{C_1, C_2, \cdots, C_m\} = \prod_{j=1}^{n} S_j$, where
       $S_i \cdot S_j \triangleq \{z_i^{C_{i1}} z_j^{C_{j1}}, z_i^{C_{i2}} z_j^{C_{j2}}, \cdots, z_i^{C_{im}} z_j^{C_{jm}}\}(mod\ N^2)$
   where the modular arithmetic $mod\ N^2$ is applied to each component.
4: return $R$
```

Algorithm 7 PBD Response Retrieval RR (Client)

```
Input: $R = \{C_1, C_2, \cdots, C_m\}, sk = \{p, q\}$
Output: $B$
1: Let $\lambda = lcm(p-1, q-1)$
2: Compute $B = \{Decrypt(C_1, sk), Decrypt(C_2, sk), \cdots, Decrypt(C_m, sk)\}$, where
       $Decrypt(C_i, sk) = \frac{(C_i^\lambda(mod\ N^2) - 1)/N}{(g^\lambda(mod\ N^2) - 1)/N}(mod\ N)$
   for i=$1, 2, \cdots, m$, the decryption algorithm is based on the Paillier cryptosystem [35].
3: return $B$
```

To download the block B_i from the n blocks, the client $\mathcal{C}$ runs our PBD protocol with the server $\mathcal{S}$, composed of three algorithms, Query Generation (QG), Response Generation (RG), and Response Retrieval (RR), as described in Algorithms 5-7.

Note that p, q, g, N, sk, pk in Algorithms 5-7 are different from those in Algorithms 1-4.

Theorem 2 (Correctness) Our PBD protocol is correct. In other words, for any cloaking region CR, the number n of blocks, the index i $(1 \leq i \leq n)$, and the public key PK, we have $B_i = \text{RR}(R, sk)$ holds, where $(Q, sk) = \text{QG}(i, CR, n, PK)$, $R = \text{RG}(Q, PK)$, $CR = \{B_1, B_2, \cdots, B_n\}$ and $B_j = \{C_{j1}, C_{j2}, \cdots, C_{jm}\}$ for $j = 1, 2, \cdots, n$.

Proof According to Algorithm 7, we have

$$\text{RR}(R, sk) = \{Decrypt(C_1, sk), \cdots, Decrypt(C_m, sk)\}.$$

To prove the theorem, we only need to prove that $C_{i1} = Decrypt(C_1, sk)$. According to Algorithms 5-7, we can obtain

$$\begin{aligned} C_1 &= \prod_{j=1}^{n} z_j^{C_{j1}} \\ &= (g^0 r_1^N)^{C_{11}} (g^0 r_2^N)^{C_{21}} \cdots (g^1 r_n^N)^{C_{i1}} \cdots (g^0 r_n^N)^{C_{n1}} \\ &= g^{C_{i1}} (\prod_{j=1}^{n} r_j^{C_{j1}})^N (mod\ N^2), \end{aligned}$$

Because a Paillier encryption of m is $g^m r^N (mod\ N^2)$, we can see that C_1 is an encryption of C_{i1}, i.e.,

$$C_{i1} = Decrypt(C_1, sk).$$

In the same way, we can prove that $C_{ij} = Decrypt(C_j, sk)$ for $j = 2, 3, \cdots, m$ and then $B_i = \text{RR}(R, sk)$. Therefore, the theorem is proved. △

In our PBD protocol, the server $\mathcal{S}$ can charge the client $\mathcal{C}$ according to the size of CR and n. The cost for the client to download the block is linear in the size of CR/n.

Our construction is similar to that of [9], [26]. The work of [26] is based upon the cryptosystem of [19], which is homomorphic over the group $\mathbb{Z}_2$, having ciphertext group $\mathbb{Z}_N$ for a large composite N. The approach by [9] is also based upon the Paillier cryptosystem [35]. The difference is that our construction aims to download a block while their construction aims to retrieve a single data element only.

Like [9], the client may partition CR into n blocks according to the dimensions of the data warehouse, such that $n = n_1 n_2 \cdots n_d$, where n_i is the total number of values of the ith dimension and d is the total number of the dimensions of CR. For the dimension with n_i different values, the client runs Algorithm 1 to generate a sub-query $Q_i = \{z_{i1}, z_{i2}, \cdots, z_{in_i}, N_i\}$, where only the ciphertext corresponding to the interested block is the encryption of 1 and $N_i^2 < N_{i+1}$ for $i = 1, 2, \cdots, d-1$. Then the client sends a query $Q = \{CR, (n_1, Q_1), (n_2, Q_2), \cdots, (n_d, Q_d)\}$ to the server, which runs Algorithm 2 for all sub-queries in series to generate a response. In the end, the client decrypts the response to obtain the block of interest from the encrypted data warehouse. For details, please refer to [9].

4.3 Private OLAP Operations

Typical OLAP operations include roll-up (performing aggregation by climbing up a concept hierarchy), drill-down (the reverse of roll-up), slice (performing a selection on one dimension, resulting in a sub-cube), dice (performing a selection on two or more dimensions, resulting in a sub-cube), and pivot (rotating the data axes in view in order to provide an alternative presentation of the data).

For private OLAP operations, we consider two cases as follows: Case 1 - the client $\mathcal{C}$ has not been provided the encrypted data warehouse $E(D)$; Case 2 - the client $\mathcal{C}$ has been provided the encrypted data warehouse $E(D)$ or has downloaded an encrypted block B_i from the server $\mathcal{S}$.

In Case 1, the client $\mathcal{C}$ may request the server $\mathcal{S}$ to perform some OLAP operations which are not sensitive to the privacy of queries. For example, roll-up from months to quarters along the time dimension, drill-down from states to cities along to the location dimension, and pivot the time and the location dimensions. For slice and dice operations which are most sensitive to the privacy of queries, the client $\mathcal{C}$ may run our PBD protocol to download a slice or a dice of the encrypted data warehouse $E(D)$ from the server $\mathcal{S}$.

In Case 2, when the client $\mathcal{C}$ has been provided $E(D)$, it can perform slice, dice or pivot operation on $E(D)$ as he does on the original data cube D because the dimension values in $E(D)$ are in plaintext. It is obvious that the sub-cube obtained by slice, dice or pivot operation on the encrypted data cube $E(D)$ takes a form of encryption of the sub-cube obtained by the same operation on the original data cube D.

For a roll-up operation on $E(D)$, without loss of generality, we consider summarising a measure x_i along the jth dimension from a concept $y_j \in \{a_{11}, a_{12}, ...\}$ to a higher concept $Y_j \in \{A_1, A_2, \cdots, A_\rho\}$, where $A_i = \{a_{i1}, a_{i2}, \cdots, a_{i\ell}\}$. For example, roll-up from months $=\{Jan., Feb., Mar., Apr., May, Jun., Jul., Aug., Sep., Oct., Nov., Dec.\}$ to quarters $=\{Q_1, Q_2, Q_3, Q_4\}$, where $Q_1 = \{Jan., Feb., Mar.\}$, $Q_2 = \{Apr., May, Jun\}$, $Q_3 = \{Jul., Aug., Sep.\}$ and $Q_4 = \{Oct., Nov., Dec.\}$.

Our roll-up operation on $E(D)$ climbing the jth dimension from a concept $\{a_{11}, a_{12}, \cdots\}$ to a higher concept $\{A_1, A_2, \cdots, A_\rho\}$ can be described in Algorithm 8.

Algorithm 8 Roll-Up (Client)

```
Input: $E(D) = (E(x_1), E(x_2), \cdots, E(x_t))_{y_1,\cdots,y_j,\cdots,y_d}, PK, j, y_j, \{a_{11}, a_{12}, \cdots\}, Y_j, \{A_1, A_2, \cdots, A_\rho\}$ where for $i = 1, 2, \cdots, t$, $A_i = \{a_{i1}, a_{i2}, \cdots, a_{i\ell}\}$
Output: $E(D)^* = (E(X_1), E(X_2), \cdots, E(X_t))_{y_1,\cdots,Y_j,\cdots,y_d}$
1: For $i = 1$ to $\rho$
2: { Compute
       $(E(X_1), E(X_2), \cdots, E(X_t))_{y_1,\cdots,A_i,\cdots,y_d} = \prod_{k=1}^{\ell} (E(x_1), E(x_2), \cdots, E(x_t))_{y_1,\cdots,a_{ik},\cdots,y_d}$
3: } //End of For
4: return $(E(X_1), E(X_2), \cdots, E(X_t))_{y_1,\cdots,A_i,\cdots,y_d}$ $(i = 1, 2, \cdots, \rho)$
```

Theorem 3 In Algorithm 4, given $1 \leq i \leq t$, let $X_{A_k} = E(X_i)_{(y_1,\cdots,A_k,\cdots,y_n)}$ and $x_{a_{ik}} = (x_i)_{(y_1,\cdots,a_{ik},\cdots,y_n)}$, then $Decrypt(X_{A_k}, SK) = \sum_{k=1}^{\ell} x_{a_{ik}}$.

Proof According to Algorithm 8, we have $X_{A_k} = \prod_{k=1}^{\ell} E(x_{a_{ik}})$. Due to the additionally homomorphic property of the BGN and Paillier cryptosystems, we have $X_{A_k} = E(\sum_{k=1}^{\ell} x_{a_{ik}})$. Therefore, we $Decrypt(X_{A_k}, SK) = \sum_{k=1}^{\ell} x_{a_{ik}}$. The theorem is proved. $\triangle$

Theorem 3 ensures that our roll-up operation on the encrypted data cube is correct.

If the client runs our PCR protocol to decrypt the roll-up results, he may randomly add some fake queries to hide the real dimension of his interested data.

When the client $\mathcal{C}$ has downloaded an encrypted block (or sub-cube) B_i of $E(D)$, it can perform the roll-up operation on B_i in the same way as described in Algorithm 8. We only need to change $E(D)$ to B_i.

At last, drill-down is reverse to roll-up. It can be implemented by roll-up from the base of the encrypted data warehouse $E(D)$ to any footprint.

5 Privacy Analysis

5.1 Privacy Analysis of PCR

In this section, we analyse the security of our Private Cell Retrieval protocol (PCR) in terms of the server's privacy and the client's privacy defined in Section 3.3.

The Paillier cryptosystem provides semantic security against chosen-plaintext attacks (IND-CPA). The ability to successfully distinguish the challenge ciphertext essentially amounts to the ability to decide composite residuosity. The decisional composite residuosity (DCR) problem can be described as: given a composite integer N and an integer y, decide whether y is N-residue modulo N^2 or not, i.e., whether there exists an integer x such that $y = x^N (mod\ N^2)$.

When $N = pq$ where p, q are distinct large primes, Paillier [35] has shown that the DCR problem is as hard as the integer factorisation problem. Therefore, the DCR problem is believed to be intractable.

Since the definition for the server's privacy is the same as the semantic security of the Paillier scheme and thus we have

Theorem 4 Under the decisional composite residuosity (DCR) assumption (i.e., the DCR problem is hard), the server in our Paillier-based PCR protocol has (semantic) data privacy.

Based on the definition of client's privacy, we consider the following game:

(1) Given the public/private key pair (PK, SK) of the Paillier cryptosystem, the adversary $\mathcal{A}$ chooses two different ciphertexts C_1 and C_2, and then sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Query Generation (QG) to obtain $(Q_b, s_b) = \mathsf{QG}(C_b, PK)$. According to Algorithm 6, we have $Q_b = C_b^{s_b} (mod\ N^2)$, where s_b is randomly chosen from $\{1, 2, \cdots, N\}$ and known to the challenger $\mathcal{C}$. Then Q_b is sent back to $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of Q_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0, 1\}$.

Due to the randomness of s_b chosen by the client in Step 2 of the game, we have

Theorem 5 In our Paillier-based PCR protocol, the client has (semantic) query privacy.

5.2 Privacy Analysis of PBD

In this section, we analyse the security of our Private Block Download protocol (PBD). Because our PBD protocol operates on the encrypted data only and does not involve any decryption by the server, it is obvious that our PBD protocol has server's privacy. Thus we only need to analyse client's privacy for our PBD protocol.

Based on the definition of client's privacy in Section 3.3, after the challenger $\mathcal{C}$ specifies a cloaking region CR in the encrypted data warehouse $E(D)$ and the total number n of blocks in CR and provides the adversary $\mathcal{A}$ with CR and n, we consider the following game:

(1) The adversary $\mathcal{A}$ selects n blocks with the same dimensions from $E(D)$ according to CR and chooses two different blocks B_0 and B_1 of the n blocks and then sends B_0, B_1 to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random $b \in \{0, 1\}$, and executes the Query Generation (QG) to obtain $(Q_b, sk) = \mathsf{QG}(b, CR, n, PK)$, where $Q_b = \{CR, n, z_1, z_2, \cdots, z_n, (g, N)\}$ and $sk = \{p, q\}$ as described in Algorithm 5. Then $\mathcal{C}$ sends Q_b back to $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment the code of Q_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0, 1\}$.

In Step 2, $z_1, z_2, \cdots, z_n$ are encryptions of either 0 or 1 by the Paillier cryptosystem with the public key $pk = (g, N)$ generated by $\mathcal{C}$. If the adversary $\mathcal{A}$ can win the game with non-negligible advantage, we can use the adversary $\mathcal{A}$ to break the Paillier cryptosystem.

Theorem 6 Under the decisional composite residuosity (DCR) assumption (i.e., the DCR problem is hard), the client in our Private Block Download (PBR) protocol has (semantic) query privacy.

Proof Assume that the adversary $\mathcal{A}$ can win the game with non-negligible advantage ϵ. Now we use the adversary $\mathcal{A}$ to break the Paillier cryptosystem.

Suppose that we are given the public key $pk = (g, N)$ of the Paillier cryptosystem, we challenge two plaintexts 0 and 1 and we are randomly given the encryption of either 0 or 1, denoted as β. Given CR and n, we construct the query Q_b by letting $z_i = \beta$ and $z_j = Encrypt(0, pk)$ for $j \neq i$. When β is the encryption of 1, Q_b stands for a real query. In this case, the probability of the adversary $\mathcal{A}$ in guessing b correctly is $1/2 + \epsilon$, where ϵ is non-negligible. When β is the encryption of 0, Q_b contains encryptions of 0s only and is independent of b. In this case, the adversary $\mathcal{A}$ can only guess randomly and the probability of the adversary $\mathcal{A}$ in guessing b correctly is $1/2$. When the adversary $\mathcal{A}$ guesses b correctly, we conclude that β is the encryption of 1 and 0 otherwise. In this way, our probability to break the semantic security of the Paillier cryptosystem is $1/2 \cdot 1/2 + 1/2 \cdot (1/2 + \epsilon) = 1/2 + \epsilon/2$. In other words, we break the semantic security of the Paillier cryptosystem with a non-negligible advantage.

However, under the decisional composite residuosity (DCR) assumption, the Paillier cryptosystem has semantic security against IND-CPA. Therefore, there does not exist an adversary $\mathcal{A}$ who can win the game with non-negligible advantage and PBD protocol has the (semantic) query privacy. $\triangle$

5.3 Privacy Analysis of Multiple Queries with PBD/PCR

In this section, we analyse the client's privacy when both PBD and PCR are run in succession.

Based on the definition of client's privacy for multiple queries in Section 3.3, we consider the following game between an adversary (the server) $\mathcal{A}$ and a challenger $\mathcal{C}$ as follows.

(1) Give an encrypted data cube $E(D)$ and the public/private key pair (PK, SK) of the server, the adversary $\mathcal{A}$ chooses two different sequences of ciphertexts or blocks of ciphertexts, $A_{0,1}, A_{0,2}, \cdots, A_{0,\ell}$ and $A_{1,1}, A_{1,2}, \cdots, A_{1,\ell}$, where $A_{i,j}$ is either a ciphertext or a block of ciphertexts. When $A_{i,j}$ is a block of ciphertexts, $A_{0,j}$ and $A_{1,j}$ are two of n_j blocks of ciphertexts with the same dimensions. Then $\mathcal{A}$ sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Query Generation (QG) on $A_{b,j}$ to obtain $(Q_{b,j}, s_{b,j}) = \mathsf{QG}(A_{b,j}, PK)$ (in the case of PCR) or $(Q_{b,j}, sk_{b,j}) = \mathsf{QG}(b, CR_j, n_j, PK)$ (in the case of PBD), where $j = 1, 2, \cdots, \ell$ and $s_{b,j}$ and $sk_{b,j}$ are the secrets of the challenger $\mathcal{C}$. and then sends $Q_{b,1}, Q_{b,2}, \cdots, Q_{b,\ell}$ back to the adversary $\mathcal{A}$.
In the case of the BGN-based PCR protocol, $Q_{b,j} = e(A_{b,j}, g)e(g, g)^{s_{b,j}}e(g, h)^{r_{b,j}}$ if $C_{b,j} \in \mathbb{G}$ or $Q_{b,j} = C_{b,j}e(g, g)^{s_{b,j}}e(g, h)^{r_{b,j}}$ if $C_{b,j} \in \mathbb{G}_1$, where $s_{b,j}, r_{b,j}$ are randomly chosen from $\{1, 2, \cdots, N\}$ and known to the challenger $\mathcal{C}$.
In the case of the Paillier-based PCR, $Q_{b,j} = C_{b,j}^{s_{b,j}}$ where $s_{b,j}$ is randomly chosen from $\{1, 2, \cdots, N\}$ and known to the challenger $\mathcal{C}$.
In the case of PBD, $Q_{b,j} = \{CR_j, n_j, z_{b,1,j}, \cdots, z_{b,n,j}, (g, N)\}$ where $z_{b,k,j} = Encrypt(1, pk) = g^1 r_{b,k,j}^N$ if k is the index of the block $A_{b,j}$ or $z_{b,k,j} = Encrypt(0, pk) = g^0 r_{b,k,j}^N (mod\ N^2)$ if k is not the index of the block $A_{b,j}$, and $r_{b,k,j}$ is randomly chosen for $\mathbb{Z}_{N^2}^*$. We assume that $sk_{b,j} = \{p, q\}$ for all b and j.

(3) The adversary $\mathcal{A}$ can experiment with the code of $Q_{b,1}$, $Q_{b,2}, \cdots, Q_{b,\ell}$ in an arbitrary non-black-box way, and finally outputs $b' \in \{0, 1\}$.

In Step 3, all queries $Q_{b,1}, Q_{b,2}, \cdots, Q_{b,\ell}$ look independent to the adversary $\mathcal{A}$ because $s_{b,j}, r_{b,j}$ (in the case of PCR) and $r_{b,k,j}$ (in the case of PBD) are randomly chosen by the challenger $\mathcal{C}$.

Theorem 7 Under the decisional composite residuosity (DCR) assumption (i.e., the DCR problem is hard), the client has (semantic) query privacy for multiple queries based on Definition 3.

Proof Assume that the adversary $\mathcal{A}$ can win the above game with a probability p and the multiple queries are composed of ℓ_1 PCR queries and ℓ_2 PBD queries. Since all queries look independent to the adversary, the adversary wins the game in two cases - winning the game by querying ℓ_1 PCR or winning the game by querying ℓ_2 PBD. Assume that the probability of the adversary winning the game by querying a single PCR is p_1 and the probability of the adversary winning the game by querying a single PBD is p_2, then $p \leq p_1\ell_1/\ell + p_2\ell_2/\ell$, i.e., $p - 1/2 \leq (p_1 - 1/2)\ell_1/\ell + (p_2 - 1/2)\ell_1/\ell$. If the adversary $\mathcal{A}$ can win the above game with a non-negligible advantage $\epsilon = p - 1/2$, then either $p_1 - 1/2$ or $p_2 - 1/2$ is non-negligible, which is in contradiction with either Theorem

5 or 6. Therefore, the advantage of the adversary in winning the above game is negligible and the theorem is proved. △

6 Performance Analysis

6.1 Performance Analysis of Paillier-Based PCR

Our Private Cell Retrieval (PCR) protocol is composed of Query Generation, Response Generation and Response Retrieval. Before the client and the server can run the PCR protocol, the server needs to encrypt the entire data warehouse D in Algorithm 1 and distribute it to the client. This initialisation costs $O(|D|)$ computation complexity and $O(|D|)$ communication complexity in the server, and $O(|D|)$ communication complexity in the client. This initialisation happens only once. Then the client and the server can run our PCR protocol repeatedly.

In the query generation (Algorithm 2), the client generates a query (Q, s) with one exponentiation $C^s(mod\ N^2)$ and sends the result to the server.

In the response generation (Algorithm 3), the server receives Q and generates a response R with one modular exponentiation and then returns the result to the client. Note that g^λ in the decryption can be pre-computed.

In the response retrieval (Algorithm 4), after receiving R, the client retrieves m with modular multiplications only.

6.2 Performance Analysis of PBD

To make our PCR protocol feasible, our PBD protocol allows the client to download a part of the encrypted data warehouse without revealing to the server which part is downloaded.

In the query generation (Algorithm 5) of our PBD protocol, the client needs to generate a public and private key pair for the Paillier cryptosystem once. The public and private keys can be used repeatedly. Therefore, the public and private key pair can be pre-generated. In addition, the client needs to compute n ciphertexts $z_1, z_2, \cdots, z_n$ under the Paillier cryptosystem for the query Q, where n is the number of blocks in the cloaking region CR. With reference to Algorithm 5, the client needs to compute $O(n)$ modular exponentiations.

In the response generation (Algorithm 6), the server needs to compute $O(nm)$ modular exponentiations, where m is the number of the ciphertexts in each block . Note the computation of modular multiplications can be omitted in comparison to the computation of modular exponentiations.

In the response retrieval (Algorithm 7), the client needs to decrypt m ciphertexts $C_1, C_2, \cdots, C_m$ under the Paillier cryptosystem, which amounts to $O(m)$ modular exponentiations.

In addition, the client sends $O(n)$ ciphertexts to the server while the server sends back $O(m)$ ciphertexts to the client. Therefore, the total communication complexity is $O(n) + O(m)$, which is much less than $O(nm)$ ciphertexts (the size of the cloaking region).

Our PBD protocol is particularly efficient in communication. It needs only $(O(n) + O(m))/O(nm) \approx 1/n$ communication of the entire CR download. In the case where $n = 100$, $m = 100,000$ and the ciphertext size is 2,402 bits, our PBD protocol needs 240.44M bits of communications, while the entire CR download needs 24.02G bits of communications. In addition, the client's computation complexity $O(n) + O(m)$ is much less than the server's computation complexity $O(nm)$. In practice, the server has more powerful computation capability than the client. Our PBD protocol reflects this feature. To improve the computational efficiency, the server may run our PBD protocol in parallel on multiple computers because our PBD protocol can support parallel computing. In the case where there are ℓ computers running in parallel in the server, the computation complexity of each computer in the server is $O(nm/\ell)$ only.

6.3 Performance Analysis of Private OLAP

With our PBD protocol, the client can perform private slice and dice in the server side. The computation and communication complexities for our PBD protocol are analyzed in the last section.

The computational complexity for the client to perform slice, dice or pivot operation on the local encrypted data cube is the same as that of the operation on the original data cube. The computational complexity of the roll-up operation on the local encrypted data cube in Algorithm 8 is $O(|E(D)|)$ group multiplications, where $|E(D)|$ is the number of ciphertexts in $E(D)$. The drill-down operation can be implemented with the roll-up operation and thus its performance depends on the performance of the roll-up operation.

6.4 Comparison

Private Information Retrieval (PIR) requires that the communication complexity is less than the size of the database. We can achieve this by combining our PBD and PCR protocols, i.e., the client runs our PBD protocol to download a block of the encrypted data warehouse from the server at first and then runs our PCR protocol with the server to decrypt a cell of the block.

Assume that the cloaking region CR is the entire data warehouse and has $n = n_1 n_2 \cdots n_d$ blocks according to d dimensions of the data warehouse, the performance comparison of our PBD+PCR protocols with existing single database PIR protocols, such as [8], [9], [15], [25], [31], [32] is shown in Table 1. In Table 1, m in our protocol is the number of ciphertexts in a block. In other protocols, m is 1. From Table 1, we can see that the performance of our PBD + PCR protocols are comparable to the performance of other PIR protocols. In particular, our PBD + PCR protocols offers both the server's privacy and the client's privacy, while other PIR protocols have the client's privacy only. In addition, the existing PIR solutions cannot keep the privacy of OLAP operations for the client.

In Table 1, we also compare the performance of our PCR protocols with existing single database SPIR protocols (also known as oblivious transfer (OT) protocols), such as Naor-Pinkas protocols [31], [32]. In these protocols, the server encrypts the data $(x_1, x_2, \cdots, x_n)$ using keys $(k_1, k_2, \cdots, k_n)$ to get ciphertexts $(C_1, C_2, \cdots, C_n)$, and sends the ciphertexts to the client. The client and server then engage in a OT protocol, where the client learns a key. The client uses this key to decrypt the data.

Our PCR protocols can supports homomorphic additions or one homomorphic multiplication over the encrypted data. But NP protocols [31], [32] do not have the homomorphic property for their encrypted data.

When the client has no constraint in communication and storage, he can download from the server the whole cloaking region CR with n encrypted blocks, one of which is what the client wants to use. Whenever the client wishes to retrieve a cell from the block, he runs the Paillier-based PCR with the server. This is the most efficient solution and privacy of this solution is the same as our PBD and Paillier-based PCR protocols.

TABLE 1
Performance Comparison

Protocols	Communication Complexity	Computation Complexity
KO [25]	O($\|D\|^{\epsilon}$) any $\epsilon > 0$	client O($\|D\|^{\epsilon}$) server O($\|D\|/2$)
CMS [8]	O($\log^8 \|D\|$)	client O($\log \|D\|$) server O($\|D\|/2$)
Chang [9]	O($\|D\|^{\epsilon}$) any $\epsilon > 0$	client O($\|D\|^{\epsilon}$) server O($\|D\|/2$)
GR [15] ($N = \prod_i {p_i}^{e_i}$)	O($\log^2 \|D\|$)	client O($\sum_i e_i(\log N + \sqrt{p_i})$) [36] server O($\|D\|/2$)
Our PBD+BGN-PCR	O($(\|D\|/m)^{1/d}$)+O(m)	client O($(\|D\|/m)^{1/d}$)+O(m) +O($\sqrt{T}$) server O($\|D\|/2$)
Our PBD+Paillier-PCR	O($(\|D\|/m)^{1/d}$)+O(m)	client O($(\|D\|/m)^{1/d}$)+O(m) +O(1) server O($\|D\|/2$)
NP [31], [32]	O($\|D\|$)	client O($\log \|D\|$) server O($\log \|D\|$)
Our BGN-PCR [41]	O($\|D\|$)	client O($\sqrt{T}$) server O(1)
Our Paillier-PCR	O($\|D\|$)	client O(1) server O(1)

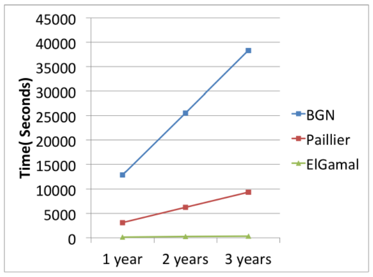

Fig. 4. Encryption Time of Initialisation

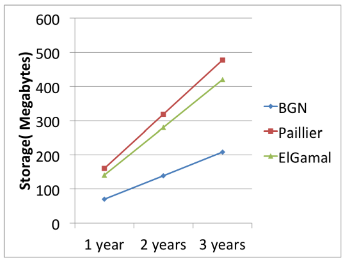

Fig. 5. Encryption Sizes of Initialisation

7 Experimental Evaluation

In order to evaluate the practice of our private data warehouse queries, we have carried out some experiments based on the TPC-DS standard [1]. This sample database is composed of many multi-dimensional data cubes and each row in a fact table represents one day. We focus on a particular sub-cube, which is centered on a store_sales table. To reduce the complexity, we limit the dimensions of this cube to date, item, store, and customer and one measure, sales_quantity.

From the data cube, we extract 5 datasets of increasing sizes: (1) one-year dataset with 531188 rows; (2) two-year dataset with 1056034 rows; (3) three-year dataset with 1590217 rows; (4) four-year dataset with 2117308 rows; (5) five-year dataset with 2647188 rows.

Our experiment is executed on a desktop machine with a Intel Core i7-2600 processor, which has a clock speed of 3.40GHz, and 16GB of RAM, and we use SQL and the C programming language.

7.1 Experimental Evaluation with PCR

To implement our BGN-based PCR protocol, we implemented the BGN cryptosystem [7], in which the elliptic curve structures $\mathbb{G}, \mathbb{G}_1$ and associated bilinear pairing e (Type A pairings) are provided by the Pairing Based Cryptography (PBC) library[2].

1. http://www.tpc.org/tpcds/default.asp
2. http://crypto.stanford.edu/pbc/

In our setting, we choose the two primes q_1 and q_2, each of which has roughly 512 bits in length, so that it is impossible to factorise N according to the current computing technology. Then we generate the public/private key pair (PK, SK) where $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h, e(g, g)^{q_1}\}$, $N = q_1 q_2$ and $SK = \{q_1\}$.

Based on the BGN cryptosystem [7], we encrypt all the measure values in the 5 datasets by Algorithm 1, where the Pailler encryption is replaced with the BGN encryption. In addition, we also experimentally evaluate Algorithm 1 with the Paillier [35] and ElGamal [14] schemes, at the same security level, and compare their performance in Fig. 4 and Fig. 5.

From Fig. 4 and Fig. 5, we can see that the initialization based on the BGN cryptosystem takes longer time than that based on other cryptosystems, but it requires less space for storing the encrypted measures. The initialization based the ElGamal cryptosystem is the fastest.

Remark. In the BGN cryptosystem, the ciphertext is a point over an elliptic curve with the x and y coordinates. The x and y coordinates satisfy an elliptic curve equation, for example, $y^2 = x^3 + x$ over a finite field F_q. Given the x-coordinate of any point, we can determine the y-coordinate by computing $y = (x^3 + x)^{2^{-1}}$, where 2^{-1} stands for the inverse of 2 $(mod\ q)$. Therefore, we need to keep the x-coordinate of a ciphertext only.

Experiment 1 (Private Cell Retrieval) Assumes that the client has a copy of the data warehouse, but all of the measures are encrypted. The client requests the server to decrypt 100 measure by running our PCR protocol with the server.

The total running time of 100 cell retrieval operations (in parallel) performed using our BGN-based PCR protocol is 10.5 seconds. If we use Paillier or ElGamal cryptosystem as the underlying encryption scheme, the total running time of 100 cell retrieval operations is 8.2 or 25.6 seconds. The Paillier-based PCR protocol has better decryption performance than other schemes.

In theory, the decryption of the Paillier scheme is independent of the size of the plaintext, but the decryption of both the ElGamal and BGN schemes depends on the size of the plaintext. This is why the Paillier scheme is more efficient than the ElGamal and BGN schemes in the decryption.

7.2 Experiment Evaluation with Private OLAP

Given an encrypted data warehouse, the slice, dice and pivot OLAP operations can be performed as in the plain data warehouse and the roll-up operation can be performed with Algorithm 8.

Experiment 2 (Private OLAP operations) Given the encrypted data warehouse, the client performs a dice operation to obtain sales_quantity per day in Store 1 from 1998 to 2002. Based on the encrypted subcube, Store_1, the client further performs a roll-up operation along the date dimension from day to month (i.e., aggregating sales_quantity for each month) with the Algorithm 8, implemented by the C programming language for successively larger datasets (1/1/98-31/12/98, 1/1/1998-31/12/99 and so on). The running times of the roll-up operation using BGN cryptosystem, along with Paillier and ElGamal cryptosystem, is illustrated in Fig. 6.

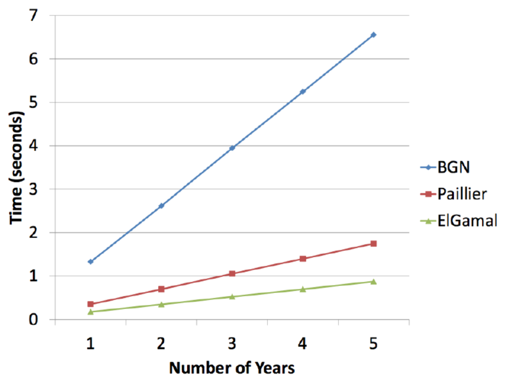

Fig. 6. Running Times of Roll-up Operation

From Fig. 6, we can see that the roll-up operation with the BGN cryptosystem is slower than the roll-up operation with the Paillier and ElGamal cryptosystems. If we perform the roll-up operation for ℓ Stores in five years, the Paillier and ElGamal cryptosystems need about 1.8ℓ and ℓ seconds, respectively. The Paillier (or ElGamal) cryptosystem is three times (or six times) faster than the BGN cryptosystem in the roll-up operation.

7.3 Experimental Evaluation with PBD

To evaluate the practical applicability of our Private Block Download (PBD) protocol, we implemented the Paillier cryptosystem [35]. With reference to [35], we generate a public/private key pair (pk, sk) for the client, where $pk = \{g, N\}$ and $sk = \{p, q, \lambda\}$ where $\lambda = lcm(p-1, q-1)$ (i.e., the least common multiple of $p-1$ and $q-1$). Based on the setting, we perform an experiment about private slice between the client and the server.

Experiment 4 (Private Block Download - Slice) Consider the subcube Store_1 created in Experiment 2. We chooses three cloaking regions: (1) 1/1/00 - 31/12/00; (2) 1/1/00 - 31/12/01; and (3) 1/1/00-31/12/02, respectively. Along with the date dimension, such cloaking regions are equally divided into: (1) 2; (2) 4; and (3) 6 slices, respectively. Assume that the client desires to download one slice from the cloaking regions with our PBD protocol. The computation and communication performance of our PBD protocol in each case is illustrated in Fig. 7 and Fig. 8, respectively.

Note that the public and private key pair of the Paillier cryptosystem for the client can be pre-generated and reused without any security concerns. In view of this, the running time of our PBD protocol in Fig. 7 is composed of the time for the client to compute n ciphertexts $z_1, z_2, \cdots, z_n$, where n is the number of blocks in the cloaking region CR, the time for the server to generate the response, and the time for the client to decrypt the response. The main component of the running time is spent by the server to compute the modular exponentiation $z_j^{C_{jk}}$ for any C_{jk} in the CR. Therefore, as shown in Fig. 7, the running time goes down with the increase of the CR. Since the computation of the exponentiation is independent of each other, the process can be executed in parallel by the server to save time. For example, if the server runs 10 computers in parallel, our PBD protocol can be speeded up by 10 times.

From Fig. 7 and Fig. 8, we can see that given a CR, the running time and communication overhead of our PBD protocol go down with the increase of the number of blocks.

In our PBD protocol, the data transmission is fast, but the computation is slow. In practice, the client can download from the server the whole cloaking region CR with n encrypted blocks, one of which is what the client wants to use. This solution has the same privacy as the PBD protocol, but this solution can significantly improve the efficiency of the private data warehouse queries. In this experiment, downloading the whole cloaking region from the server over a 100Mbps network takes less than 10 seconds.

8 Conclusion

In this paper, we have presented a new solution, based on the Paillier cryptosystem, for private cell retrieval from data warehouse after the client gets a copy of the encrypted data warehouse. Our solution allows the client to perform OLAP operations, such as roll-up, drill-down, dice, slice, pivot on the encrypted data warehouse, and then retrieve a cell from the resulting data warehouse without revealing to the server which operations are performed and which cell is retrieved. Our Paillier-based solution is more efficient than our BGN-based solution [41]. Our solution provides not only client's privacy but also server's privacy. Performance analysis and experiments show that our solution is practical for private data warehouse queries.

To enhance our private cell retrieval protocol, we also presented a solution for private block download which allows the client to download a block from the encrypted data warehouse without revealing which block is downloaded. Our private block download protocol makes our private cell retrieval protocol feasible. The performance of our private block download can be significantly improved with parallel computations. The client can execute our PBD protocol with the server multiple times. To guarantee the privacy of the client's query, the client had better choose the previous cloaking region or a new cloaking region without overlap with the previous cloaking region.

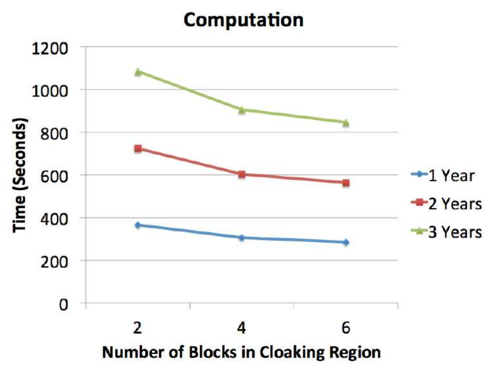

Fig. 7. Running Times of PBD

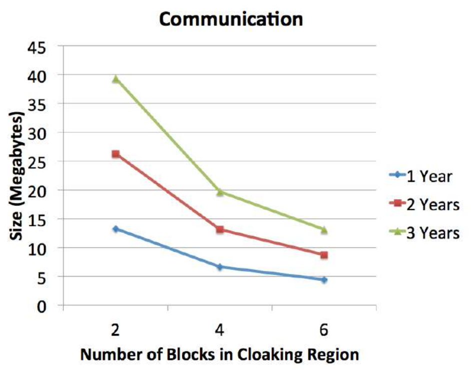

Fig. 8. Communication Overheads of PBD

References

[1] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proc. 24th International Colloquium on Automata, Languages and Programming*, pages 401-407, 1997.

[2] A. Arasu, K. Eguro, R. Kaushik, and R. Ramamurthy. Querying Encrypted Data, In *Proc. ICDE'13*, pages 1262-1263, 2013.

[3] D. Asonov. Private information retrieval - an overview and current trends. In *Proc. ECDPvA Workshop, Informatik'01*, 2001.

[4] M. A. Aufaure, N. Kuchmann-Beauger, P. Marcel, S. Rizzi and Y. Vanrompay. Predicting your next OLAP query based on recent analytical sessions. In *Proc. DaWak'13*, pages 134-145, 2013.

[5] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: a unified construction. In *Proc. 28th International Colloquium on Automata, Languages and Programming*, pages 912-926, 2001.

[6] A. Boldyreva, N. Chenette, and A. O'Neill. Order - preserving encryption revisited: Improved security analysis and alter- native solutions. In *Proc. CRYPTO'11*, pages 578-595, 2011.

[7] D. Boneh, E. Goh and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. TCC'05*, pages 325-341, 2005.

[8] C. Cachin, S. Micali and M. Stadler. Computational private information retrieval with polylogarithm communication. In *Proc. EUROCRYPT'99*, pages 402-414, 1999.

[9] Y. Chang. Single database private information retrieval with logarithmic communication. In *Proc. 9th Australasian Conference on Information Security and Privacy*, pages 50-61, 2004.

[10] B. Chor and N. Gilboa. Computational private information retrieval. In *Proc. 29th ACM Symposium on the Theory of Computing*, pages 304-313, 1997.

[11] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. 36th IEEE Symposium on Foundations of Computer Science*, pages 41-51, 1995.

[12] I. Damgard and M. Jurik. A Generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proc. PKC'01*, pages 119-136, 2001.

[13] M. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proc. EUROCRYPT'10*, pages 24-43, 2010.

[14] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31 (4): 469-472, 1985.

[15] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Proc. 31th International Colloquium on Automata, Languages and Programming*, pages 803-815, 2005.

[16] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. STOC'09*, pages 169-178, 2009.

[17] Y. Gentner, Y. Ishai, E. Kushilevitz and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. 30th ACM Symposium on Theory of Computing*, pages 151-160, 1998.

[18] G. Ghinita, P. A. Khoshgozaran, C. Shahabi and K. L. Tan. Private queries in location based services: anonymizers are not necessary. In *SIGMOD'08*, pages 121-132, 2008.

[19] S. Goldwasser and S. Micali. Probabilistic encryption. *Journal of Computer and System Science*, 28(2): 270-299, 1984.

[20] J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* 2nd Edition. Morgan Kaufmann Publishers, 2006.

[21] W. H. Inmon. *Building the Data Warehouse.* John Wiley & Sons, 1996.

[22] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. 31st ACM Symposium on the Theory of Computing*, pages 79-88, 1999.

[23] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fund. Electron. Communi. Comput. Sci.* E. 82-A(1): 11-20, 1999

[24] T. Itoh. On lower bounds for the communication complexity of private information retrieval. *IEICE Trans. Fund. Electron. Communi. Comput. Sci.* E. 84-A(1): 157-164, 2001.

[25] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computational private information retrieval. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pages 364-373, 1997.

[26] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *Proc. EUROCRYPT'00*, pages 104-121, 2000.

[27] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *Proc. 8th International Conference on Information Security*, pages 314-328, 2005.

[28] H. Lipmaa. First CPIR protocol with data-dependent computation. In *Proc. ICISC'09*, pages 193-210, 2009.

[29] A. Menezes, P. van Oorchot and S. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.

[30] S. K. Mishra and P. Sarkar. Symmetrically private information retrieval. In *Proc. INDOCRYPT'00*, pages 225-236, 2000.

[31] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. 31th ACM Symposium on Theory of Computing*, pages 245-254, 1999.

[32] M. Naor and B. Pinkas. Efficient oblivious transfer protocol. In *Proc. SODA'01*, pages 448-457, 2001.

[33] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Proc. EUROCRYPT'98*, 1998.

[34] R. Ostrovsky and W. E. Skeith III. A survey of single-database PIR: techniques and applications. In *Proc. PKC'07*, pages 393-411, 2007.

[35] P. Paillier. Public key cryptosystems based on composite degree residue classes. In *Proc. EUROCRYPT'99*, pages 223-238, 1999.

[36] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1): 106-110, 1978.

[37] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2): 120-126, 1978.

[38] N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proc. PKC'10*, pages 420-443, 2010.

[39] L. Sweeney. 2002. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10 (5): 557-570.

[40] X. Yi, M. Kaosar, R. Paulet and E. Bertino. . Single-database private information retrieval from fully homomorphic encryption. *IEEE Trans. on Knowledge and Data Eng.*, 25(5): 1125-1134,2013.

[41] X. Yi, R. Paulet, E. Bertino, G. Xu. Private data warehouse queries. In *SACMAT'13*, pages 25-36, 2013.

Summary of Changes from Conference Version

A preliminary version of this paper has been published in 18th ACM Symposium on Access Control Models and Technologies (SACMAT), Amsterdam, The Nethertands, June 12-14, 2013.

The conference version is attached to this submission. The additional content compared to the conference version includes:

1) A new model for private block download (PBD) in Section 3.2.
2) Security definitions for PBD in Section 3.3.
3) Client's privacy definition for multiple queries in Section 3.3.
4) A new private cell retrieval (PCR) protocol based on the Paillier cryptosystem (Algorithms 1-4) in Section 4.1.
5) A new PBD protocol (Algorithms 5-7) in Section 4.2.
6) Privacy analysis on our Paillier-based PCR protocol in Section 5.1.
7) Privacy analysis on our PBD protocol in Section 5.2.
8) Privacy analysis on multiple queries with PBD/PCR in Section 5.3.
9) Performance analysis on our Paillier-based PCR protocol in Section 6.1.
10) Performance analysis on our PBD protocol in Section 6.2.
11) Performance comparison in Section 6.4.
12) Experiment evaluation on our solutions in Section 7.

Private Data Warehouse Queries

Xun Yi
School of Engineering and Science
Victoria University
Melbourne, VIC 8001, Australia
xun.yi@vu.edu.au

Russell Paulet
School of Engineering and Science
Victoria University
Melbourne, VIC 8001, Australia
russell.paulet@vu.edu.au

Elisa Bertino
Department of Computer Science
Purdue University
West Lafayette, IN 47907, USA
bertino@purdue.edu

Guandong Xu
Advanced Analytics Institute
University of Technology, Sydney
Broadway NSW 2007, Australia
guandong.xu@uts.edu.au

ABSTRACT

Publicly accessible data warehouses are an indispensable resource for data analysis. But they also pose a significant risk to the privacy of the clients, since a data warehouse operator may follow the client's queries and infer what the client is interested in. Private Information Retrieval (PIR) techniques allow the client to retrieve a cell from a data warehouse without revealing to the operator which cell is retrieved. However, PIR cannot be used to hide OLAP operations performed by the client, which may disclose the client's interest. This paper presents a solution for private data warehouse queries on the basis of the Boneh-Goh-Nissim cryptosystem which allows one to evaluate any multi-variate polynomial of total degree 2 on ciphertexts. By our solution, the client can perform OLAP operations on the data warehouse and retrieve one (or more) cell without revealing any information about which cell is selected. Furthermore, our solution supports some types of statistical analysis on data warehouse, such as regression and variance analysis, without revealing the client's interest. Our solution ensures both the server's security and the client's security.

Categories and Subject Descriptors

H.2 [**Database Management**]: Security, integrity, and protection; H.2.7 [**Database Administration**]: Data warehouse and repository

General Terms

Security

Keywords

Data warehouse, OLAP, privacy, homomorphic encryption

1. INTRODUCTION

Data warehousing provides tools for business executives to systematically organise, understand, and use their data to make strategic decisions. A large number of organizations have found that data warehouses are valuable in today's competitive, fast-evolving world. In the last several years, many firms have spent millions of dollars in building enterprise-wide data warehouse. Many people feel that with competition mounting in every industry, data warehouse is the latest must-have marketing weapon - a way to keep customers by learning more about their needs [16].

A data warehouse is a **subject - oriented**, **integrated**, **time - variant** and **non - volatile** collection of data in support of management's decision making process [17]. Data warehouses are built on a multidimensional data model. This model views data in the form of a data cube. A data cube, defined by dimensions and measures, allows data to be viewed in multiple dimensions. In general, dimensions are the entities with respect to which we want to keep records. For example, a sales data warehouse may keep records of the store's sales with respect to dimensions - time, location and product. Measures are the quantities by which we want to analyse relationships between dimensions. Examples of measures for a sales data warehouse include the sales amount, the number of units sold, and the average sales amount.

In the multidimensional model, data is organised into multiple dimensions, where each dimension has multiple-levels of abstraction defined by a concept hierarchy. A concept hierarchy defines a sequence of mapping from a set of low-level concepts to high-level, more general concepts. The concept hierarchy for locations could be street, city, state, and country. This organisation provides clients with the flexibility to view data from different perspectives. A number of online analytical processing (OLAP) operations exist to materialise these different views, supporting interactive querying and analysis of the data at hand. Typical OLAP operations include: **roll-up** (aggregation by climbing up a concept hierarchy); **drill-down** (the reverse of roll-up); **slice** (a selection on one dimension, resulting in a sub-cube); **dice** (a selection on two or more dimensions, resulting in a sub-cube); **pivot** (rotating the data axes in view in order to provide an alternative presentation of the data).

Queries to the data warehouse are based on a star-net model, which consists of radial lines emanating from a central point, where each line represents a concept hierarchy of a dimension. These represent the granularities available for use by OLAP operations such as drill-down and roll-up.

In order to query the data warehouse, a user usually first requests the server to perform OLAP operations and then sends back a cell. An important issue in this simple process is represented by the privacy of the user query as a user query may reveal to the server business sensitive information about the user. For example, for a stock exchange data warehouse, the user may be an investor, who queries the data warehouse for the trend of a certain stock. He may wish to keep private the identity of the stock he is interested in. For a pharmaceutical data warehouse, the user may be a laboratory, which would like to keep private the active principles it wants to use. To protect his privacy, the user accessing a data warehouse may therefore want to perform OLAP operations and retrieve a cell without revealing any information about which cell he is interested in.

A trivial solution to the above private data warehouse query problem is for the user to download the entire data warehouse and then locally perform OLAP operations and retrieve the cell of interest. This solution is not suitable if the owner of the data warehouse wishes to make profit through data warehouse services (for example, a health care data warehouse). Usually, the user is interested in only a part of the data warehouse. Purchasing the entire data warehouse may not be an economic way to the user.

Private Information Retrieval (PIR) protocols, such as [8], do not fully address the private data warehouse query problem. A PIR protocol allows a user to retrieve a record from a database without the owner of that database being able to determine which record was selected with communication cost less than the database size. By using PIR, a user can retrieve a cell (a record) from a data warehouse (a database) without revealing any information about which cell is retrieved. However, the user cannot hide his OLAP operations to the server when he requests the server to perform the operations. These operations may reveal the user's interest. For example, when the user requests the server to perform a slice operation with respect to a location, the server can learn the user's interest in the location. It is a challenge to assure the user's privacy when performing OLAP operations.

In this paper, we give a solution for private data warehouse queries on the basis of the Boneh-Goh-Nissim cryptosystem [4]. Our basic idea is to allow the data warehouse owner to encrypt its data warehouse and distribute the encrypted data warehouse to the user who wishes to perform private data warehouse queries. The user can perform any OLAP operations on the encrypted data warehouse locally without revealing his interest. When the user wishes to decrypt a cell of the encrypted data warehouse, the user and the server run a Private Cell Retrieval (PCR) protocol jointly to decrypt the cell without revealing to the server which cell is retrieved. Assume that the serve charges the client per query, our solution allows the user to perform some statistical analysis, such as regression and variance analysis, on the encrypted data warehouse with the lowest cost.

Unlike operational databases, data warehouse is non-volatile. The data in the data warehouse is never over-written or deleted - once committed; the data is static, read-only, and retained for future reporting. It is feasible to allow the data warehouse owner to distribute the encrypted data warehouse to potential users only once and later let the users download new added data online if any.

Our solution ensures both the server's security in the sense that the server, for billing purpose, releases to the user only a data per query, and the client's security in the sense that the client does not reveal any information about his queries to the server. We have implemented our solution on an example of data warehouse and experiments have shown that our solution is practical for private data warehouse queries.

The rest of the paper is organized as follows. Related work is surveyed in Section 2. We define our model and described our solution in Section 3. The security and performance analysis is carried out in Section 4. Experiment results are shown in Section 5. Conclusions are drawn in the last section.

2. RELATED WORK

Private Information Retrieval (PIR) was firstly introduced by Chor, Goldreich, Kushilevitz, and Sudan in 1995 [8]. In their paper, they proposed a set of schemes to implement PIR through replicated databases, which provide users with information-theoretic security as long as some of the database relicas do not collude against the user. Since then, a lot of research on PIR has been done. We classify the results as follows [2, 29].

- *Theoretical Private Information Retrieval*

  "Theoretical" stands for the fact, that the user privacy is assumed to be unbreakable independently from the computational power of a cheater. Chor et al. proved, that any theoretical PIR solution has a communication with a lower bound equal to the database size [8]. Thus they relax the problem setting. They assume that there are several (instead of one) database servers, which do not communicate among each other, with the same data. This assumption makes the non-trivial theoretical PIR feasible. The basic idea is to send several queries to several databases. The queries are constructed in such a way, that they give no information to the servers about the record that user is interested in. But using the answers from the queries, the user can construct the desired record. There is also a case considered, when up to t of the servers are allowed to cooperate against the user.

  Ambainis [1] improved the results of Chor et al. [8], which led to the following non-trivial theoretical PIR solutions: (1) A k database scheme (i.e., a scheme with k identical databases non-communicating to each other), for any constant $k \geq 2$, with communication complexity $O(N^{1/(2k-1)})$; (2) A $\Theta(\log N)$ database scheme with communication complexity $O(log^2 N \cdot loglogN)$, where N is the size of the database. Further research on theoretical PIR appears in [18, 19, 20, 3].

- *Computational Private Information Retrieval*

  In order to get better communication complexity, another assumption was weakened by Chor and Gilboa [7]. "Computational" means that the database servers are presumed to be computationally bounded, i.e., under an appropriate intractability assumption, the database cannot gain information about which data element was selected by the user. For every $\epsilon > 0$, Chor and Gilboa [7] presented two computational PIR schemes with complexity $O(N^\epsilon)$.

  In the first paper on PIR [8] it was proven, that the theoretical PIR problem has no non-trivial solutions for the case of single database. Surprisingly, the substitution of an information-theoretic security with an intractability assumption achieves a non-trivial PIR protocol for single database schema [22]. Its communication complexity is $O(N^\epsilon)$ for any $\epsilon > 0$. They use an intractability assumption, described in [15]. The basic approach is to encrypt a query in such a way, that the server still can process it using special algorithms. However, the server recognizes neither the clear-text query nor the result.

The result can be decrypted only by the user. This was the first single-database protocol, where designer considers and provides database privacy. Using another intractability assumption, Cachin et al. [5] demonstrated a single database computational PIR protocol that has poly-logarithmic communication. This is an improvement in comparing to polynomial communication complexity in [22]. This result looks particular effective, because the user has to send a minimum $\log N$ bits just to address the bit he wants to retrieve in the database. A scheme with better result appears in [6, 23, 24, 12].

- *Symmetrical Private Information Retrieval*

  Symmetrical PIR is a PIR problem, where the privacy of the database is considered, i.e., a symmetrical PIR protocol must prevent the user from learning more than one record of the database during a session. Clearly, symmetrical privacy (database privacy) is a very important property for practical applications, since an efficient billing is only then possible. A symmetrical PIR protocol for single server was first proposed in [21], and for several servers it was considered in [14]. Other symmetrical PIR were later proposed in [27, 26].

In addition, Private Block Retrieval (PBR) is a natural extension of PIR in which, instead of retrieving only a single bit, the user retrieves a d-bit block that begins at an index i. PBR techniques are important for making PIR practical. Theoretic PBR was introduced in [8]. A practical PBR scheme for a single database was given by Gentry and Ramzan [12]. The security of this scheme is based on a simple variant of the Φ-hiding number-theoretic assumption by Cachin, Micali and Stadler [5]. This scheme has communication complexity $O(k+d)$ only, where $k \geq \log N$ is a security parameter that depends on the database size N and d is the bit-length of the retrieved database block.

Current PIR or PBR protocols can be only used for Private Cell Retrieval (PCR) in private data warehouse queries. They are unable to support private OLAP operations.

3. PRIVATE DATA WAREHOUSE QUERIES

3.1 Our Model

We consider a data cube D with n dimensions $y_1, y_2, \cdots y_n$ and m measures $x_1, x_2, \cdots, x_m$, denoted as

$$D = (x_1, x_2, \cdots, x_m)_{y_1, y_2, \cdots, y_n}.$$

We assume that the data cube is provided by a server S and used by clients. The server S wishes to make a profit by providing data warehouse services to clients. The clients wish to learn some knowledge from D through OLAP operations on D without revealing their interests to S.

First of all, on input a security parameter k, the server S generates its public/private key pair $\{PK, SK\}$, encrypts the data cube D into $E(D)$ with the public key PK, where the values of all measure attributes are encrypted, but the values of all dimension attributes are in plaintexts. The encrypted data cube $E(D)$ can be then released to clients.

A client $\mathcal{C}$ can either download the encrypted data cube $E(D)$ from the server's Web site or request the server to send a CD of the encrypted data cube by post. It happens only once because the data warehouse is non-volatile. For new data added into the data cube, we allow the users to download it online. The client can then perform any OLAP operation on the encrypted data cube $E(D)$ locally.

Remark In many cases, the "client" is actually an organization that then has no problem in downloading and storing the encrypted data warehouse $E(D)$ and performing OLAP operations on $E(D)$.

In order to retrieve a cell from the data cube D after several OLAP operations on $E(D)$ (i.e., to decrypt a ciphertext C from the encrypted data cube $E(D)$ after several OLAP operations on $E(D)$), the server S and the client $\mathcal{C}$ runs a Private Cell Retrieval (PCR) protocol, composed of three algorithms as follows.

(1) Query Generation (QG): Takes as input the public key PK of the server S, the ciphertext C, which is an encryption of either a measure value or a function of several measure values, (the client) outputs a query Q and a secret s, denoted as $(Q, s) = \mathsf{QG}(C, PK)$.

(2) Response Generation (RG): Takes as input the query Q and the private key SK of the server S, (the server) outputs a response R, denoted as $R = \mathsf{RG}(Q, SK)$.

(3) Response Retrieval (RR): Takes as input the public key PK of the server S, the response R and the secret s of the client, (the client) outputs a plaintext x, denoted as $x = \mathsf{RR}(R, PK, s)$.

A PCR protocol can be illustrated as in Fig. 1 and is correct if, for any security parameter k, for any ciphertext C, $Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$ holds, where $(Q, s) = \mathsf{QG}(C, PK)$ and $R = \mathsf{RG}(Q, SK)$.

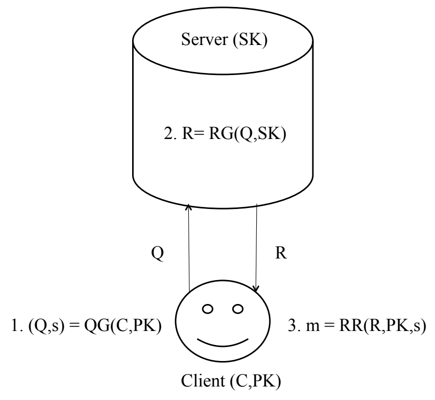

Figure 1: Private Cell Retrieval

The security of the PCR protocol involves the server's security and the client's security. Intuitively, the serve S wishes to release only one measure value to the client $\mathcal{C}$ each time when the client sends a query. Meanwhile, the client $\mathcal{C}$ does not wish to reveal to the server which cell is retrieved.

Formally, the server's security can be can be defined with a game as follows.

Given a data cube D and the public key PK of the server, consider the following game between an adversary (the client) $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) Given the public key PK of the server, the adversary $\mathcal{A}$ chooses two different values m_1, m_2 of two measure attributes and sends them to $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and encrypts m_b to obtain $C_b = Encrypt(m_b, PK)$, and then sends C_b back to $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of C_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0,1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be

$$\mathsf{Adv}_{\mathcal{A}}(k) = |\mathsf{Pr}(b' = b) - 1/2|.$$

Definition 1 (Server's Security Definition). In a PCR protocol, the data warehouse server has semantic security if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

Remark. Server's security ensures that the client cannot decrypt any ciphertext in the encrypted data cube $E(D)$ without the help of the server.

Next, we formally define the client's security with a game as follows.

Give an encrypted data cube $E(D)$ and the public/private key pair (PK, SK) of the server, consider the following game between an adversary (the server) $\mathcal{A}$, and a challenger $\mathcal{C}$. The game consists of the following steps:

(1) Given the public/private key pair (PK, SK) of the server, the adversary $\mathcal{A}$ chooses two different ciphertexts C_1 and C_2, and then sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0,1\}$, and executes the Query Generation (QG) to obtain $(Q_b, s_b) = \mathsf{QG}(C_b, PK)$, where s_b is the secret of the challenger $\mathcal{C}$, and then sends Q_b back to $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of Q_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0,1\}$.

The adversary wins the game if $b' = b$ and loses otherwise. We define the adversary $\mathcal{A}$'s advantage in this game to be

$$\mathsf{Adv}_{\mathcal{A}}(k) = |\mathsf{Pr}(b' = b) - 1/2|.$$

Definition 2 (Client's Security Definition). In a PCR protocol, the client has semantic security if for any probabilistic polynomial time (PPT) adversary $\mathcal{A}$, we have that $\mathsf{Adv}_{\mathcal{A}}(k)$ is a negligible function, where the probability is taken over coin-tosses of the challenger and the adversary.

Remark. Client's security ensures that the server cannot tell what information the client has retrieved from the data cube $E(D)$.

3.2 Private Cell Retrieval

Based on our model, we give a construction of a PCR protocol which allows the client to retrieve a measure value in a cell without revealing the measure and cell attributes to the server.

Our protocol is built on the BGN homomorphic encryption scheme [4] (please refer to Appendix). The data server S generates and publishes its public key $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h_1, e(g,g)^{q_1}\}$, and keeps its private key $SK = \{q_1\}$ secret.

Remark: Slightly different from the BGN scheme, we replace h with h_1 in the public key (please refer to Appendix) and include $e(g,g)^{q_1}$ as a public parameter. It does not affect the security of the BGN scheme because the discrete logarithm problem of determining the private key q_1 from $e(g,g)^{q_1}$ is hard, where q_1 is large prime. We publish $e(g,g)^{q_1}$ so that the client $\mathcal{C}$ can obtain the decryption privately (please refer to Algorithm 4).

Before releasing the data cube to clients, the data warehouse server S runs the Initialisation algorithm to encrypt the data cube D to $E(D)$, as described in Algorithm 1.

After obtaining the encrypted data cube $E(D)$, if a client $\mathcal{C}$ wishes to retrieve a measure value in a cell, in other words, to decrypt a ciphertext C in a cell, the client $\mathcal{C}$ and the server $\mathcal{S}$ run our Private Cell Retrieval protocol, composed of three algorithms, Query Generation (QG), Response Generation (RG), and Response Retrieval (RR), as described in Algorithms 2-4.

Algorithm 1 Initialisation (Server)

```
Input: $D = (x_1, x_2, \cdots, x_m)_{y_1, y_2, \cdots, y_n}, PK$
Output: $E(D) = (E(x_1), E(x_2), \cdots, E(x_m))_{y_1, y_2, \cdots, y_n}$
1: Let $E(D) = D$
2: For each measure value $x = (x_i)_{y_1, y_2, \cdots, y_m}$, where $1 \le i \le m$ and $(y_1, y_2, \cdots, y_m) \in DD$ (dimension domain) .
3: Pick a random integer $r$ from $\{1, 2, \cdots, N\}$
4: Compute $z = Encrypt(x, PK) = g^x h^r$ and replace $(x_i)_{y_1, y_2, \cdots, y_n}$ with $z$, denoted as $(E(x_i))_{y_1, y_2, \cdots, y_n}$.
5: return $E(D)$
```

Algorithm 2 Query Generation QG (Client)

```
Input: $C, PK$
Output: $Q, s$
1: Pick two random integers $s, r$ from $\{1, 2, \cdots, N\}$
2: If $C \in \mathbb{G}$, compute $Q = e(C, g)e(g, g)^s h_1^r$
3: If $C \in \mathbb{G}_1$, compute $Q = Ce(g, g)^s h_1^r$
4: return $(Q, s)$
```

Algorithm 3 Response Generation RG (Server)

```
Input: $Q \in \mathbb{G}_1, SK = q_1$
Output: $R$
1: Compute $R = Q^{SK}$
2: return $R$
```

Algorithm 4 Response Retrieval RR (Client)

```
Input: $R, PK, s$
Output: $m$
1: Compute $R' = R/(e(g, g)^{q_1})^s$
2: Compute $m = \log_{e(g,g)^{q_1}} R'$ with Porland's lambda method [25].
3: return $m$
```

Theorem 1 (Correctness) Our PCR protocol is correct. In other words, for any security parameter k, for any ciphertext C,

$$Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$$

holds, where $(Q, s) = \mathsf{QG}(C, PK)$ and $R = \mathsf{RG}(Q, SK)$.

Proof In case of the ciphertext $C \in \mathbb{G}$, we assume that $C = g^{m'} h^{r'}$. With reference to Appendix, we have $Decrypt(C, SK) =$

m'. In addition,

$$\begin{aligned} R &= \mathsf{RG}(Q, SK) = Q^{SK} \\ &= (e(C,g)e(g,g)^s h_1^r)^{q_1} \\ &= e(g^{m'}h^{r'}, g)^{q_1} e(g,g)^{q_1 s} h_1^{q_1 r} \\ &= e(g^{q_1 m'} h^{q_1 r'}, g) e(g,g)^{q_1 s} h_1^{q_1 r} \\ &= (e(g,g)^{q_1})^{m'} e(g,g)^{q_1 s} \end{aligned}$$

In case of the ciphertext $C \in \mathbb{G}_1$, we assume that $C = e(g,g)^{m'} h_1^{r'}$. With reference to Appendix, we have $Decrypt(C, SK) = m'$. In addition,

$$\begin{aligned} R &= \mathsf{RG}(Q, SK) = Q^{SK} \\ &= (Ce(g,g)^s h_1^r)^{q_1} \\ &= (e(g,g)^{q_1})^{m'} e(g,g)^{q_1 s} h_1^{q_1(r+r')} \\ &= (e(g,g)^{q_1})^{m'} e(g,g)^{q_1 s} \end{aligned}$$

Therefore, $R' = R/e(g,g)^{q_1 s} = (e(g,g)^{q_1})^{m'}$ and we have $m = \log_{e(g,g)^{q_1}} R' = m'$, i.e., $Decrypt(C, SK) = \mathsf{RR}(R, PK, s)$. The theorem is proved. $\triangle$

3.3 Private OLAP Operations

Typical OLAP operations include roll-up (performing aggregation by climbing up a concept hierarchy), drill-down (the reverse of roll-up), slice (performing a selection on one dimension, resulting in a sub-cube), dice (performing a selection on two or more dimensions, resulting in a sub-cube), and pivot (rotating the data axes in view in order to provide an alternative presentation of the data).

After obtaining the encrypted data cube $E(D)$, a client $\mathcal{C}$ can perform drill-down, slice, dice or pivot operation on $E(D)$ as he does on the original data cube D because the dimension values in $E(D)$ are in plain. It is obvious that the sub-cube obtained by slice, dice or pivot operation on the encrypted data cube $E(D)$ takes a form of encryption of the sub-cube obtained by the same operation on the original data cube D.

For a roll-up operation on $E(D)$, without loss of generality, we consider summarising a measure x_i along the j-th dimension from a concept $y_j \in \{a_1, a_2, ...\}$ to a higher concept $Y_j \in \{A_1, A_2, ...\}$, where for any a_s, there is A_t such that $a_s \in A_t$. Our roll-up operation on $E(D)$ is described in Algorithm 5.

Algorithm 5 Roll-Up (Client)

```
Input: $E(D) = (E(x_1), E(x_2), \cdots, E(x_m))_{y_1,\cdots,y_j,\cdots,y_n}$, $PK$,
       $\{A_1, A_2, ...\}$
Output: $E(D)^* = (E(x_1), E(x_2), \cdots, E(x_m))_{y_1,\cdots,Y_j,\cdots,y_n}$
 1: Let $E(D)^* = (E(0), E(0), \cdots, E(0))_{y_1,\cdots,Y_j,\cdots,y_n}$
 2: For each encrypted measure value $x = (E(x_i))_{y_1,\cdots,y_j,\cdots,y_m}$
    in $E(D)$, where $1 \le i \le m$ and $y_j \in \{a_1, a_2, \cdots\}$
 3: If $y_j = a_s \in A_t$ and $X = (E(x_i))_{y_1,\cdots,Y_j,\cdots,y_n}$ in $E(D)^*$
    where $Y_j = A_t$, let $Z = xX$ and replace $X$ with $Z$ in the cell
    $(y_1, \cdots, Y_j, \cdots, y_n)$ of $E(D)^*$.
 4: return $E(D)^*$
```

Theorem 2 In Algorithm 5, given $1 \le i \le m$, let

$$X_{A_t} = E(x_i)_{(y_1,\cdots,Y_j,\cdots,y_n)}$$

where $Y_j = A_t$ and $x_{a_s} = (x_i)_{(y_1,\cdots,y_j,\cdots,y_n)}$ where $y_j = a_s$, then $Decrypt(X_{A_t}, SK) = \sum_{a_s \in A_t} x_{a_s}$.

Proof According to Algorithm 5, we have

$$X_{A_t} = \prod_{a_s \in A_t} E(x_{a_s}).$$

Due to the homomorphic property of the BGN cryptosystem, we obtain

$$X_{A_t} = E(\sum_{a_s \in A_t} x_{a_s}).$$

Therefore,

$$Decrypt(X_{A_t}, SK) = \sum_{a_s \in A_t} x_{a_s}$$

The theorem is proved. $\triangle$

Theorem 2 ensures that our roll-up operation on the encrypted data cube is correct.

3.4 Private Statistical Analysis

Our data cube is encrypted by the BGN cryptosystem. As shown in Appendix, the BGN cryptosystem has an additive homomorphism. In addition, the bilinear map allows for one multiplication on encrypted values. As a result, the BGN cryptosystem supports arbitrary additions and one multiplication (followed by arbitrary additions) on encrypted data. This property in turn allows the evaluation of multi-variate polynomials of total degree 2 on encrypted values.

In view of this, we are able to perform those statistical analyses on the data cube in private, which involves the evaluation of multi-variate polynomials of total degree 2 on encrypted values, e.g., regression and variance analysis.

Remark. Most practical homomorphic cryptosystems, such as RSA [32], ElGamal [11], Goldwasser-Micali [15], Damgard-Jurik [9] and Paillier [30] schemes, provide only one homomorphism, either addition, multiplication, or XOR. They cannot be used to evaluate multi-variate polynomials of total degree 2 on encrypted values. Some statistical analysis requires to compute multi-variate polynomials of total degree 2. Although fully homomorphic encryption techniques [13, 33, 10] can be used to evaluate multi-variate polynomials of any degree, the state-of-the-art is still impractical in applications because the ciphertext size and computation time increase sharply as one increases the security level. So far, the BGN cryptosystem [4] is the only practical encryption scheme which can evaluate multi-variate polynomials of total degree 2 on encrypted values.This is why we choose the BGN cryptosystem as our underlying encryption scheme.

Let $f(x_1, x_2, \cdots, x_\ell)$ be a ℓ-variate polynomial of total degree 2. For a purpose of statistical analysis, a user wishes to compute $f(a_1, a_2, \cdots, a_\ell)$ in private, where $a_1, a_2, \cdots, a_\ell$ are measure values in the data cube D. Given the encrypted data cube $E(D)$, the user obtains the encryptions of $a_1, a_2, \cdots, a_\ell$, denoted as $E(a_1), E(a_2), \cdots, E(a_\ell)$ and runs Algorithm 6.

Algorithm 6 Private Evaluation (Client, Server)

```
Input: $f, E(a_1), E(a_2), \cdots, E(a_\ell), PK$
Output: $f(a_1, a_2, \cdots, a_\ell)$
 1: Client computes $C = f(E(a_1), E(a_2), \cdots, E(a_\ell))$
 2: Client and Server run Algorithms 2-4
 3: Client obtains $m = RR(R, PK, s)$
 4: return $m$
```

Theorem 3 In Algorithm 6, $m = f(a_1, a_2, \cdots, a_\ell)$.

Proof Because the BGN cryptosystem allows the evaluation of multivariate polynomials of total degree 2 on encrypted values and the degree of the function f is less than 2, we have that

$$C = f(E(a_1), E(a_2), \cdots, E(a_\ell)) = E(f(a_1, a_2, \cdots, a_\ell)).$$

Based on Theorem 1, we have that

$$m = Decrypt(C, SK) = f(a_1, a_2, \cdots, a_\ell).$$

The theorem is proved. $\triangle$

Theorem 3 ensures that our private evaluation is correct.

4. SECURITY AND PERFORMANCE ANALYSIS

4.1 Security Analysis

In this section, we analyse the security of our Private Cell Retrieval protocol (PCR) in terms of the server's security and the client's security defined in Section IV. We consider the server's security at first.

In our scenario, the server wishes to make profit through data warehouse services. The business model is most likely that the server charges the client per query. In other word, the server reveals one measure value only in each client query. In order to prevent the client from knowing all data in the data warehouse without paying queries, the server encrypts the data warehouse with the BGN cryptosystem [4], where the decryption key is known the server only.

The security of the BGN cryptosystem is built on the subgroup decision problem: With reference to Appendix, let $k \in \mathbb{Z}^+$ and let $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$ be a tuple produced by $KeyGen(k)$ where $N = q_1 q_2$. Given $(N, \mathbb{G}, \mathbb{G}_1, e)$ and an element $x \in \mathbb{G}$, output 1 if the order of x is q_1 and output 0 otherwise; that is, without knowing the factorization of the group order N, decide if an element x is in a subgroup of $\mathbb{G}$.

We say that $KeyGen(k)$ satisfies the subgroup decision assumption if for any polynomial time algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in solving the subgroup decision problem,

$$|Pr(\mathcal{A}(N, \mathbb{G}, \mathbb{G}_1, e, x) = 0) - Pr(\mathcal{A}(N, \mathbb{G}, \mathbb{G}_1, e, x) = 1)|,$$

is a negligible function in k.

In [4], it has been shown that the BGN scheme is semantically secure if $KeyGen(k)$ satisfies the subgroup decision assumption.

Theorem 4 If $KeyGen(k)$ satisfies the subgroup decision assumption in the BGN scheme, the server in our Private Cell Retrieval (PCR) protocol has the semantic security.

Proof Please refer to [4] for the proof that the BGN scheme is semantic security if $KeyGen(k)$ satisfies the subgroup decision assumption.

Slightly different from the BGN scheme, we replace h with h_1 in the public key and include $e(g, g)^{q_1}$ as a public parameter. Because $h_1 = e(g, h)$, the replacement does not affect the security of the BGN scheme. In addition, it is hard to determine q_1 from $e(g, g)^{q_1}$ because the discrete logarithm is hard, and $e(g, g)^{q_1}$ does not help to solve the subgroup decision problem at all, i.e., to decide if $x^{q_1} = 1$ given an element x. Therefore, the definition for the server's security is the same as the semantic security of the BGN scheme and the theorem is proved. $\triangle$

Next, we analyse the client's security. Based on the definition of client's security, we consider the following game:

(1) Given the public/private key pair (PK, SK) of the BGN cryptosystem, the adversary $\mathcal{A}$ chooses two different ciphertexts C_1 and C_2, and then sends them to the challenger $\mathcal{C}$.

(2) The challenger $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$, and executes the Query Generation (QG) to obtain $(Q_b, s_b) = \mathsf{QG}(C_b, PK)$. According to Algorithm 2, if $C \in \mathbb{G}$,

$$Q_b = e(C_b, g)e(g, g)^{s_b} h_1^{r_b};$$

if $C \in \mathbb{G}_1$,

$$Q_b = C_b e(g, g)^{s_b} h_1^{r_b},$$

where s_b, r_b are randomly chosen from $\{1, 2, \cdots, N-1\}$ and known to the challenger $\mathcal{C}$. Then Q_b is sent back to $\mathcal{A}$.

(3) The adversary $\mathcal{A}$ can experiment with the code of Q_b in an arbitrary non-black-box way, and finally outputs $b' \in \{0, 1\}$.

Theorem 5 The client in our Private Cell Retrieval (PCR) protocol has the semantic security.

Proof In Step 2 of the above game, the ciphertext C_b is blinded by random $e(g, g)^{)s_b} h_1^{r_b}$. Without knowledge of random s_b, r_b, the adversary $\mathcal{A}$ cannot tell which ciphertext is blinded even if $\mathcal{A}$ can apply the decryption key SK on C_b in Step 3 to obtain $R_b = e(g, g)^{q_1 m_b} e(g, g)^{q_1 s_b}$ where $m_b = Decrypt(C_b, SK)$. In view of this, the adversary $\mathcal{A}$'s advantage in this game ($\mathsf{Adv}_{\mathcal{A}}(k) = |\mathsf{Pr}(b' = b) - 1/2|$) is negligible. Therefore, the theorem is proved. $\triangle$

Remark. In Algorithm 2, if $C \in \mathbb{G}$, the client generates the query Q in $\mathbb{G}$ by letting $Q = Cg^s h^r$ where s, r are randomly chosen from $\{1, 2, \cdots, N-1\}$ and sends the query Q to the server. But this may leak to the server the client's intention, such that retrieving a cell or performing statistical analysis. Therefore, in order to keep the client's intention private, the client has to generate the query Q in $\mathbb{G}_1$ no matter whether $C \in \mathbb{G}$ or $C \in \mathbb{G}_1$ as in Algorithm 2.

4.2 Performance Analysis

The core of our solution is our Private Cell Retrieval (PCR) protocol, composed of Query Generation, Response Generation and Response Retrieval. Before the client and the server can run the PCR protocol, the server is required to encrypt the whole data warehouse D in Algorithm 1 and distribute it to the client. This initialisation costs $O(|D|)$ computation complexity and $O(|D|)$ communication complexity in the server, and $O(|D|)$ communication complexity in the client. This initialisation happens only once. Then the client and the server can run our PCR protocol any number of times.

In the query generation (Algorithm 2) of our PCR protocol, the client generates a query (Q, s) with at most two exponentiations in $\mathbb{G}_1$ and one pairing, and sends a group element of $\mathbb{G}_1$ to the server.

In the response generation (Algorithm 3), the server receives a group member of $\mathbb{G}_1$ and generates a response R with one exponentiation in $\mathbb{G}_1$ and then replies a group element of $\mathbb{G}_1$ to the client.

In the response retrieval (Algorithm 4), after receiving a group element of $\mathbb{G}_1$, the main time of the client is spent on determining the discrete logarithm $m = \log_{e(g,g)^{q_1}} R'$ with Porland's lambda method [25]. The computation complexity of Porland's lambda method is $\sqrt{T}$ where T is the upbound of m.

Computation of exponentiations and pairings and communications of group elements of $\mathbb{G}_1$ can be very fast. Thus, the main running time of our PCR protocol is $O(\sqrt{T})$. If T is around 2^{32}, the computation complexity is around $2^{16} = 65536$.

Next, we analyse the performance of private OLAP operations. The computation complexity for the client to perform drill-down, slice, dice, drill-down or pivot operation on the encrypted data cube is the same as that of the same operation on the original data cube. In Algorithm 5, assume that the domain for the dimension y_j includes λ different values, then the computation complexity of the roll-up operation is $O(\lambda)$ group multiplications, which can be done very quickly.

At last, we analyse the performance of our private statistical analysis. In Algorithm 6, to evaluate a ℓ-variate polynomial $f(x_1, x_2, \cdots, x_\ell)$ of total degree 2 at a point $(a_1, a_2, \cdots, a_\ell)$, the client and the server need jointly to run our PCR protocol once. For this evaluation, the client and the server can also run our PCR protocol ℓ times to retrieve $a_1, a_2, \cdots, a_\ell$ at first and then the client computes $f(a_1, a_2, \cdots, a_\ell)$ locally. Assume that the upbound of x_i (where $1 \leq i \leq \ell$) is T, then the upbound of $f(x_1, x_2, \cdots, x_\ell)$ is about T^2. In this case, the main running time of Algorithm 6 is about $O(T)$ while the main running time of ℓ PCR protocols is about $O(\ell\sqrt{T})$, usually less than $O(T)$. However, the client in Algorithm 6 needs to pay once while running ℓ PCR protocols needs to pay ℓ times. Therefore, our private statistical analysis Algorithm 6 has the lowest cost.

To balance the cost and the running time for private statistical analysis, the client may retrieve a part of $(a_1, a_2, \cdots, a_\ell)$ and then run Algorithm 6. The cost and running time are inversely proportional. We can see that if the client wishes to perform statistical analysis on the data warehouse with less cost, he has to spend more time to get the result. If the client wishes to perform statistical analysis on the data warehouse with less time, he has to pay more to get the result.

To the best of our knowledge, our solution is only one to provide private OLAP operations. In our solution, our private cell retrieval (PCR) protocol is essentially a private information retrieval (PIR) protocol. Unlike existing PIR protocols, such as [21, 5, 12], our PCR protocol needs to communicate the encrypted data warehouse in the initialisation to enable private OLAP operations. This happens only once. Without considering the initialisation, the performance comparison of our PCR protocol with some single database PIR protocols are listed in TABLE 1.

Table 1: Performance Comparison

Protocols	Comm. Complexity	Comp. Complexity
KO[21]	O($\|D\|^\epsilon$) any $\epsilon > 0$	client O($\|D\|^\epsilon$) server O($\|D\|/2$)
CMS[5]	O($\log^8 \|D\|$)	client O($\log \|D\|$) server O($\|D\|/2$)
GR[12] $(N = \prod_i {p_i}^{e_i})$	O($\log^2 \|D\|$)	client O($\sum_i e_i(\log N + \sqrt{p_i})$[31]) server O($\|D\|/2$)
Our PCR	O(1)	client O($\sqrt{T}$) server O(1)

From TABLE 1, we can see that our PCR protocol is more efficient than other single database PIR protocols in terms of communication if we do not consider the initialisation. In addition, only our solution supports private OLAP operations.

Compared with a centralised data warehouse which supports OLAP operations, our solution has two advantages as follows:

- A centralised data warehouse cannot protect the privacy of OLAP operations required by the client even if PIR may be used to prevent the server from knowing the final cell retrieved by the client. Our solution can protect the privacy of both OLAP operations performed by the client and the final cell retrieved by the client.
- A centralised data warehouse is inefficient when multiple clients concurrently perform OLAP operations in the server and run PIR with the server. Our solution is distributed and the client can perform OLAP operations in his local computer and only run our efficient PCR with the server.

Remark. We should point out that our solution may not be suitable for operational databases which need to update their data frequently. This will require our solution to run initialisation many times and leads the performance of our solution worse than others. Our solution is in particular suitable for data warehouse where the data is non-volatile. In this scenario, our solution needs to run initialisation only once.

5. EXPERIMENTAL EVALUATION

In order to evaluate the practice of our solution, we have done some experiments on the Oracle global data warehouse example[1], which has four dimensions, Channel, ShipTo, Product and Time, and a units fact table storing three measures, units, sales and cost. The date cube keeps 9 years sale history data and contains about 300,000 cells. Our experiment is executed on a desktop machine with a Intel Core i7-2600 processor, which has a clock speed of 3.40GHz, and 16GB of RAM, and we use SQL and C programming language.

First of all, we implemented the BGN cryptosystem [4], in which the elliptic curve structures $\mathbb{G}, \mathbb{G}_1$ and associated bilinear pairing e are provided by the Pairing Based Cryptography (PBC) library[2]. For the public/private key pair (PK, SK) where $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h_1, e(g, g)^{q_1}\}$, $N = q_1 q_2$, $h_1 = e(g, u^{q_2})$ and $SK = \{q_1\}$, we use the values in Table 2.

In our setting, we choose the two primes q_1 and q_2, each has roughly 512 bits in length, so that it is impossible to factorize N according to the current computing technology.

Based on the BGN cryptosystem, we encrypt all measure values in the units fact table. This initialisation takes about 5 hours. The size of the original data cube is 45 Mbytes while the size of the encrypted data cube becomes 850 Mbytes.

Remark In practice, a data warehouse is a very sparse multi- dimensional data set. In this case, the size of the encrypted data cube can be significantly reduced because only the measures with values need to be encrypted.

Based on the encrypted data cube $E(D)$, we have done four experiments described in what follows. The goal of these experiments is to determine the actual times required by various OLAP operations, from the most simple, that is, retrieving a single cell, to the most complex ones, such as performing regression analysis and variance analysis.

Experiment 1 (Private Cell Retrieval) Consider the ciphertext C in Table 3 which is the encrypted value of a cell the user wants to retrieve. To do so, the client generates a query $(Q, s) = QG(C, PK)$ with values of Q and s shown in Table 3, and sends Q to the server. The server generates a response $R = RG(Q, SK)$ as shown in Table 3 and turns R to the client.

The client computes $m = RR(R, PK, s)$ with Porland's lambda method, where m =3346.

[1]http://www.oracle.com/technetwork/database/options/olap/global-11g-readme-082667.html
[2]http://crypto.stanford.edu/pbc/

Table 2: Setting

Para.	Values
q_1	21075418640760410231092518815985907107218754 972770446796531020693656493360387346119630326 264278648095178013266650636409358246554306844 97131351494956236085 7
q_2	12770757865052888311329533459098171896872893 787872540228494605554106287432987166747651481 138929282665123628771740975444782201562035179 4491888424971146430 83
N	26914906836577326184182404167133398412265122 920763120895499121041552231662267127763420808 884301441386036774318686152457720001737158446 392728638727302997344285045745026304548018580 650052785473080254387997180737945903664821079 571787820745590170923086457597068802014571888 5793966526448525040987835358478405002131
g	[38541583386952104107838945807346922321315262 760669641124899551348857051123127875141089007 714527824427443188616396621912530181137147121 870447696068628604002854494442009563963141580 756926191397532999397829406565934259561055970 554703419949372001563073197258958319877077753 216741702156041831655844141890780432157126 7,9 480911976241807747934632097048398180200105998 822744599035552243464272721936069407066922848 681186534355607954535020561134603030182187722 681209428254633017341515467050549483142767178 688953392003318854120308972805480677016226334 643239789167325213173963299635951154235542201 237755220067092164079566743266180947221]
u	[90355113191707349033650321671064819233411791 554488948056631073957107032184057077540247156 969053761302738958171477136366680144692611719 400895154756559355801930222886350399605246802 012080359890355376422384293752890166573180008 042710290122021244649546918197580253304536519 279950176817945174866553861242002236175118,91 234402775643178015535429018890332467573384174 078776314525487458132229222432297506474819321 295819916644881701149670224771561523459777244 699718715893247838335015592417414766946680054 568360496527404452082429881562821921617122620 787126291583946026460700670166393134322784351 346342103872802765111089380627867938185]

Table 3: Experiment 1

Para.	Values
C	[21522238738461525457804247592603098239082392 038785063750325433774364005380080152889464771 046093100420206695664478201112198250352239173 376948379934274411021748381200445180102603667 502868789777076704050793816507371488678285357 974126104931073455246768043976272808520171912 948493082807139048394220120865039895414075 1,3 803401062802803611669710650042637313843290526 205704651819069769663874847208344009914491314 498017794637941714616376720513725601967496618 671574197331105775429251809783142282868934923 403194455486949263780893751731609669875213865 342483211597115040375577714048719922339756050 150435462595411512244945603617747945032 90]
Q	[13724215451079649073033016471841055766946184 455864722510205774752889594028721018584041778 968542667134181195335136107613784925093970189 671595118693407286662937609153435259508435516 120051646035966939254144117626700470282727608 621000507342875403371028321077317865134553805 221483458454854197574101488933997181617089,36 789190810624682660133959786033453519971380335 040480398414400575822156629834206450594065923 235201468469214716537177903387409527279919477 906928896600272638643968376015906361250408092 092671451402556606180306493098211303244025485 500508011733225235070033030481564959308782835 852683713705923144227097233354009986678 2]

s	18885767480755584952853279609689701392631490 3
	570009472779352231629993166112778876777630402
	393044880461058951359315772953414254388265799
	018043444401815792645824517581283228316321701
	047479133908428806317280841136455777666483401
	759509633043124148231096623706520495844956120
	643045136913607186633855095825607139394

R	[33324365871965840806598200562956850561308599 602140306018341714961813900816384117482223946 335082780484339957711954583092246571786099078 383822644273391553757106540887649473021231157 533482944211035652605376724676507874759043352 212103371943683688353291709453168918442266250 865883638219437764600010671859246570929590 6,6 793981002937657883003772968985115860421274050 707946258490137107897735879402078461477528407 917014419131282645454009201265043911194356556 804859002976782405306798948607938220165139491 772181244832765171489970171197466768678129027 664013605467300628414621290211719800812523241 481892911105514786006870381361360003625 2]

The total running time of this cell retrieval operation performed using our PCR protocol is about 1.14 seconds.

Experiment 2 (Private OLAP operations) Given the encrypted data warehouse, the client performs a slice operation implemented by the following SQL query:

```
CREATE VIEW Catalogues AS
SELECT ShipTo, Product, Time, units, sales, cost
FROM units_fact_table
WHERE Channel="CAT"
```

where "CAT" stands for Catalogues. The resulting Catalogues view is a 3-dimension subcube, from which the client further performs a slice operation implemented by the following SQL query,

```
CREATE VIEW Mouse AS
SELECT ShipTo, Time, units, sales, cost
FROM Catalogues
WHERE Product="MOUSE"
```

The resulting Mouse view is a 2-dimension subcube, from which the client further performs a roll-up operation along the ShipTo dimension from customer to all (i.e., aggregating three measures units, sales and cost for all customers), and then a roll-up operation along the time dimension from month to year (i.e., aggregating three measures units, sales and cost from months into years), with Algorithm 5 implemented by the C programming language.

At last, the client obtains a subcube with only the Time dimension Time, which takes values in the set $\{0, 1, \cdots, 9\}$.

The total running time for the above sequential OLAP operations is less than 3 minutes.

Remark We can speed up our roll-up operation by parallel computation. For example, if we allocate the task for the roll-up operation to multiple computers and run 5 computers in parallel, the total running time for the above experiment can be reduced from 3 minutes to 36 seconds.

Experiment 3 (Private Regression Analysis) The sequence of the OLAP operations in Experiment 2 restricts and shapes the data warehouse so that it is ready for a regression analysis, by which the client would like to investigate the relationship between the number of sold mouse units and the time.

Consider the 1-dimension subcube that resulted from the sequential OLAP operations in Experiment 2, and assume that the units measure takes values $E(Y_1), E(Y_2), \cdots, E(Y_n)$ in years $X_1, X_2, \cdots, X_n$, respectively.

A simple regression analysis is to determine b_0 and b_1 in the linear equation $Y_i = b_0 + b_1 X_i$, where the formulas for the least squares estimates are

$$\begin{aligned} b_1 &= \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sum_{i=1}^{n}(X_i - \overline{X})^2} \\ b_0 &= \overline{Y} - b_1\overline{X} \end{aligned}$$

where $\overline{X} = \frac{\sum_{i=1}^{n} X_i}{n}$ and $\overline{Y} = \frac{\sum_{i=1}^{n} Y_i}{n}$.

Given $X_1, X_2, \cdots, X_n$ and $E(Y_1), E(Y_2), \cdots, E(Y_n)$, to compute b_0, b_1, the client gets $\prod_{i=1}^{n} E(Y_i) = E(\sum_{i=1}^{n} Y_i)$ decrypted by our PCR protocol and then computes $\overline{X}$ and $\overline{Y}$. Next, let $[\overline{X}]$, $[\overline{Y}]$ be the round results of $\overline{X}, \overline{Y}$ (note that the number of sold units is positive integer), and let

$$Z = \prod_{i=1}^{n}(E(Y_i)/E([\overline{Y}]))^{X_i - [\overline{X}]}.$$

Based on the homomorphic property of the BGN cryptosystem, we have that

$$\begin{aligned} Z &= \prod_{i=1}^{n}(E(Y_i - [\overline{Y}])^{X_i - [\overline{X}]} \\ &= \prod_{i=1}^{n} E((X_i - [\overline{X}])(Y_i - [\overline{Y}])) \\ &= E(\sum_{i=1}^{n}(X_i - [\overline{X}])(Y_i - [\overline{Y}])) \\ &= E(b_1 \sum_{i=1}^{n}(X_i - [\overline{X}])^2) \end{aligned}$$

Then the client gets Z decrypted by our PCR protocol and then computes $b_1 = Decrypt(Z, SK)/\sum_{i=1}^{n}(X_i - [\overline{X}])^2$ and $b_0 = \overline{Y} - b_1\overline{X}$.

By the above private regression analysis, we obtain the linear equation

$$Y = 9407.33 - 658.08X.$$

This result is very close to the actual linear equation $Y = 9407.67$ $-658.08X$ obtained by performing the regression analysis on the plain data warehouse. In addition, our private regression analysis needs two decryptions only.

Remark. The difference between two linear equations is due to the round operation.

Experiment 4 (Private Variance Analysis) Consider the 1-dimension subcube that resulted from the sequential OLAP operations in Experiment 2, i.e., $X_1, X_2, \cdots, X_n$ (years) and $E(Y_1), E(Y_2), \cdots, E(Y_n)$ (encrypted units), and suppose that the client would like to compute the variance for units measure in private.

The variance v^2 can be computed as follows:

$$\begin{aligned} v^2 &= \frac{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}{n} \\ &= \frac{\sum_{i=1}^{n}(nY_i - n\overline{Y})^2}{n^3} \\ &= \frac{\sum_{i=1}^{n}(nY_i - \sum_{j=1}^{n} Y_j)^2}{n^3} \end{aligned}$$

From $E(Y_1), E(Y_2), \cdots, E(Y_n)$, the client can obtain $E(nY_i - \sum_{j=1}^{n} Y_j) = E(Y_i)^n / \prod_{j=1}^{n} E(Y_j)$. Let

$$Z = e(E(nY_i - \sum_{j=1}^{n} Y_j), E(nY_i - \sum_{j=1}^{n} Y_j)),$$

where e denotes the pairing operation (please refer to Appendix). Based on the homomorphic property of the BGN cryptosystem, we have

$$\begin{aligned} Z &= \prod_{i=1}^{n} e(E(nY_i - \sum_{j=1}^{n} Y_j), E(nY_i - \sum_{j=1}^{n} Y_j)) \\ &= \prod_{i=1}^{n} \mathcal{E}((nY_i - \sum_{j=1}^{n} Y_j)^2) \\ &= \mathcal{E}(\sum_{i=1}^{n}(nY_i - \sum_{j=1}^{n} Y_j)^2) \end{aligned}$$

where $\mathcal{E}$ denotes the BGN encryption over $\mathbb{G}_1$ (please refer to Appendix).

Next, the client gets Z decrypted by our PCR protocol and computes $v^2 = Decrypt(Z, SK)/n^3$.

By the above private variance computation, the client obtains the variance $v^2 = 3212337.56$ for units measure with only one decryption. This result is the same as the actual variance by performing the variance analysis on the plain data warehouse.

Remark. If our underlying encryption scheme were the ElGamal scheme [11] or Paillier scheme [30] instead of the BGN scheme [4], the private variance analysis in Experiment 4 would have needed 9 decryptions instead of 1 decryption.

6. CONCLUSION

In this paper, we have presented a solution for private data warehouse queries. Our solution allows the client to perform OLAP operations, such as roll-up, drill-down, dice, slice, pivot, and then retrieve a cell from the resulted data warehouse without revealing to the server what operations are performed and what cell is retrieved. In particular, our solution allows the client to perform some statistical analysis on the data warehouse with the lowest cost if the server charges the client per query.

Our solution provides not only the client's security but also the server's security. Performance analysis and experiments have shown that our solution is practical for private data warehouse queries.

So far, our solution only allows the client to privately perform statistical analyses which can be algebraically expressed as a polynomial of degree at most 2 on the data warehouse, such as regression and variance analysis. Our future work will extend our solution so that the client can privately perform statistical analyses which cannot be algebraically expressed as a polynomial, such as min, max and count.

7. REFERENCES

[1] A. Ambainis. Upper bound on the communication complexity of private information retrieval. In *Proc. 24th International Colloquium on Automata, Languages and Programming*, pp. 401-407, 1997.

[2] D. Asonov. Private information retrieval - an overview and current trends. In *Proc. ECDPvA Workshop, Informatik'01*, 2001.

[3] A. Beimel and Y. Ishai. Information-theoretic private information retrieval: a unified construction. In *Proc. 28th International Colloquium on Automata, Languages and Programming*, pp. 912-926, 2001.

[4] D. Boneh, E. Goh and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proc. TCC'05*, pp. 325-341, 2005.

[5] C. Cachin, S. Micali and M. Stadler. Computational private information retrieval with polylogarithm communication. In *Proc. EUROCRYPT'99*, pp. 402-414, 1999.

[6] Y. Chang. Single database private information retrieval with logarithmic communication. In *Proc. 9th Australasian Conference on Information Security and Privacy*, pp. 50-61, 2004.

[7] B. Chor and N. Gilboa. Computational private information retrieval. In *Proc. 29th ACM Symposium on the Theory of Computing*, pp. 304-313,1997.

[8] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *Proc. 36th IEEE Symposium on Foundations of Computer Science*, pp.41-51, 1995.

[9] I. Damgard and M. Jurik. A Generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proc. PKC'01*, pages 119-136, 2001.

[10] M. Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan. Fully homomorphic encryption over the integers. In *Proc. EUROCRYPT'10*, pages 24-43, 2010.

[11] T. ElGamal. A public-key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31 (4): 469ÃŘ472, 1985.

[12] C. Gentry and Z. Ramzan. Single-database private information retrieval with constant communication rate. In *Proc. 31th International Colloquium on Automata, Languages and Programming*, pp. 803-815, 2005.

[13] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proc. STOC'09*, pages 169-178, 2009.

[14] Y. Gentner, Y. Ishai, E. Kushilevitz and T. Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. 30th ACM Symposium on Theory of Computing*, pp. 151-160, 1998.

[15] S. Goldwasser S. and Micali (1984) Probabilistic encryption, Journal of Computer and System Science, 28(2), pp. 270-299.

[16] J. Han and M. Kamber, *Data Mining: Concepts and Techniques.* 2nd Edition. Morgan Kaufmann Publishers, 2006.

[17] W. H. Inmon, *Building the Data Warehouse.* John Wiley & Sons, 1996.

[18] Y. Ishai and E. Kushilevitz. Improved upper bounds on information theoretic private information retrieval. In *Proc. 31st ACM Symposium on the Theory of Computing*, pp. 79-88, 1999.

[19] T. Itoh. Efficient private information retrieval. *IEICE Trans. Fund. Electron. Communi. Comput. Sci.* E. 82-A(1), pp. 11-20, 1999.

[20] T. Itoh. On lower bounds for the communication complexity of private information retrieval. *IEICE Trans. Fund. Electron. Communi. Comput. Sci.* E. 84-A(1), pp. 157-164, 2001.

[21] E. Kushilevitz and R. Ostrovsky. Replication is not needed: single database, computational private information retrieval. In *Proc. 38th IEEE Symposium on Foundations of Computer Science*, pp. 364-373, 1997.

[22] E. Kushilevitz and R. Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *Proc. EUROCRYPT'00*, pp. 104-121, 2000.

[23] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *Proc. 8th International Conference on Information Security*, pp. 314-328, 2005.

[24] H. Lipmaa. First CPIR protocol with data-dependent computation. In *Proc. ICISC'09*, pages 193-210, 2009.

[25] A. Menezes, P. van Oorchot and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1997.

[26] S. K. Mishra and P. Sarkar. Symmetrically private information retrieval. In *Proc. INDOCRYPT'00*, pp. 225-236, 2000.

[27] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. 31th ACM Symposium on Theory of Computing*, pp. 245-254, 1999.

[28] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *Proc. EUROCRYPT'98*, 1998.

[29] R. Ostovsky and W. E. Skeith III. A survey of single-database PIR: techniques and applications. In *Proc. PKC'07*, pages 393-411, 2007.

[30] P. Paillier. Public key cryptosystems based on composite

degree residue classes. In *Proc. EUROCRYPT'99*, pages 223-238, 1999.

[31] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24(1): 106-110, 1978.

[32] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21 (2): 120ÃŘ126, 1978.

[33] N. Smart and F. Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proc. PKC'10*, pages 420-443, 2010.

APPENDIX

A. BONEH-GOH-NISSIM CRYPTOSYSTEM

We introduce the Boneh-Goh-Nissim cryptosystem [4] in this section.

A.1 Bilinear Group

We use the following notations:

1. $\mathbb{G}$ and $\mathbb{G}_1$ are two (multiplicative) cyclic groups of finite order n.
2. g is a generator of $\mathbb{G}$.
3. e is a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_1$. In other words, for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}$, we have $e(u^a, v^b) = e(u, v)^{ab}$. We also require that $e(g, g)$ is a generator of $\mathbb{G}_1$.

We sat that $\mathbb{G}$ is a bilinear group if a group $\mathbb{G}_1$ and a bilinear map as above exist.

A.2 Boneh-Goh-Nissim Encryption Scheme

Boneh-Goh-Nissim encryption scheme, BGN scheme by brevity, resembles the Paillier [30] and the Okamoto-Uchiyama [28] encryption schemes. The three algorithms making up the scheme are described as follows:

A.2.1 *Key Generation* $KeyGen(k)$

Given a secure parameter $k \in \mathbb{Z}^+$, run $KeyGen(k)$ to obtain a tuple $(q_1, q_2, \mathbb{G}, \mathbb{G}_1, e)$. Let $N = q_1 q_2$. Pick up two random generators g, u from $\mathbb{G}$ and set $h = u^{q_2}$. Then h is a random generator of the subgroup of $\mathbb{G}$ of order q_1. The public key is $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h\}$. The private key $SK = q_1$.

A.2.2 *Encryption* $Encrypt(m, PK)$

Assume the message space consists of integers in the set $\{0, 1, \cdots, T\}$ with $T < q_2$. We encrypt bits in which case $T = 1$. To encrypt a message m using the public key PK, pick a random r from $\{1, 2, \cdots, N\}$ and compute

$$C = g^m h^r \in \mathbb{G} \tag{1}$$

Output C as the ciphertext.

A.2.3 *Decryption* $Decrypt(C, SK)$

To decrypt a ciphertext C using the private key $SK = q_1$, observe that

$$C^{q_1} = (g^m h^r)^{q_1} = (g^{q_1})^m$$

To recover the message m, it suffices to compute the discrete logarithm of C^{q_1} to the base g^{q_1}. Since $0 \le m \le T$, this takes expected time $O(\sqrt{T})$ using Polland's lambda method [25].

A.3 Homomorphic Properties

The BGN scheme is clearly additively homomorphic. Let $PK = \{N, \mathbb{G}, \mathbb{G}_1, e, g, h\}$ be a public key. Given two ciphertexts $C_1, C_2 \in \mathbb{G}$ of messages $m_1, m_2 \in \{0, 1, \cdots, T\}$ respectively, anyone can create a uniformly distributed encryption of $m_1 + m_2 (mod\ N)$ by computing the product $C = C_1 C_2 h^r$ for a random r in $\{1, 2, \cdots, N-1\}$.

More importantly, anyone can multiply two encrypted messages once using the bilinear map. Let $g_1 = e(g, g)$ and $h_1 = e(g, h)$, then g_1 is of order N and h_1 is of order q_1. There is some (unknown) $\alpha \in \mathbb{Z}$ such that $h = g^{\alpha q_2}$. Suppose that we are given two ciphertexts $C_1 = g^{m_1} h^{r_1} \in \mathbb{G}$ and $C_2 = g^{m_2} h^{r_2} \in \mathbb{G}$. To build an encryption of the product $m_1 m_2 (mod\ N)$, (1) pick a random $r \in \mathbb{Z}_N$, and (2) let $C = e(C_1, C_2) h_1^r \in \mathbb{G}_1$. Then

$$\begin{aligned} C &= e(C_1, C_2) h_1^r \\ &= e(g^{m_1} h^{r_1}, g^{m_2} h^{r_2}) h_1^r \\ &= e(g^{m_1 + \alpha q_2 r_1}, g^{m_2 + \alpha q_2 r_2}) h_1^r \\ &= e(g, g)^{(m_1 + \alpha q_2 r_1)(m_2 + \alpha q_2 r_2)} h_1^r \\ &= e(g, g)^{m_1 m_2 + \alpha q_2 (m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2)} h_1^r \\ &= e(g, g)^{m_1 m_2} h_1^{r + m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2} \end{aligned}$$

where $r + m_1 r_2 + m_2 r_1 + \alpha q_2 r_1 r_2$ is distributed uniformly in $\mathbb{Z}_N$. Thus C is a uniformly distributed encryption of $m_1 m_2 (mod\ N)$, but in $\mathbb{G}_1$ rather than $\mathbb{G}$. We note that the BGN scheme is still additively homomorphic in G_1.