International Journal of Networks and Communications 2012, 2(6): 138-141
DOI: 10.5923/j.ijnc.20120206.01

Analyzing the Order Factor of Query Matrix in Private Information Retrieval Protocol for Outsourced Databases

Sunil B. Mane[1,*], Pradeep K. Sinha[2]

[1]Computer Engineering & Information Technology Department, College of Engineering, Pune, M.S., India
[2](High Performance Computing), Center for Development & Advance Computing, Pune, M.S., India

Abstract In this paper, we present analysis of non-concurrent model of fast single-database Private Information Retrieval (PIR) scheme for maintaining data privacy with data confidentiality using encryption algorithm like AES for Outsourced Database Service (ODBS) Model. In this model encryption method maintains data confidentiality and a utility that help to create PIR reply which maintains data privacy. Single-database PIR schemes are generally unusable because of its expensiveness from computational point of view. This model suggests use of Graphics Processing Unit (GPU) commodity to process database. This model will help to achieve practical implementation of single-database PIR scheme which uses encryption algorithm to maintain data confidentiality as secondary goal and it serves to create database reply to maintain data privacy as primary goal. This paper shows some real time results that proves need of concurrency in PIR algorithms. Our paper describes the use of lattice based single-database PIR protocol with GPU as a processing unit to achieve high speed-up and hence, the performance of the system.

Keywords Outsourced Database, Private Information Retrieval, User Privacy, Data Privacy

1. Introduction

A PIR scheme is a protocol in which user retrieves a record or set of records out of *n* records from the database which has been outsourced by hiding the contents from service providers or database administrators. In order to preserve data privacy replicas of the database have been stored at the service provider end. Since service provider does not know the result of the query has calculated from which copy of the database. Another way is to allow a user to retrieve privately an element of a non-replicated database known as single-database PIR schemes.

In single-database PIR schemes data privacy is related to the intractability of mathematical problem, instead of being based on the assumption that different replicas exists and do not collude against their users[1][2][3].

PIR schemes usually require the an enormous amount of computational power, but considering the huge number of applications these protocols have, it is important to develop practically implemented protocols that provide acceptable performances for as many applications as possible[1].

A major issue with single-database PIR schemes is that, they are computationally expensive. Indeed, in order to answer a query, the database must process all of its records.

If in a given protocol, it doesn't process some set of records, the database administrator will learn that, the user is not interested in them. This would reveal to the database administrator about partial information on which record the user is interested in, and therefore it is not as private as downloading the whole database and retrieving locally the desired entry.

The computational cost for a server replying to a PIR query is therefore linear on the database size. Moreover, number-theoretic schemes have a very expensive cost per bit multiplication over a large modulus in the database. This limits both the database size and the throughput shared by the users.

2. Overall Design of Proposed Model

In a PIR protocol, when a user wants to retrieve an element of index *i* from a database, he will use a PIR query generation algorithm with input *i* and send the resulting query to the database as shown in Figure1. In this paper we are focusing on the two security issues with respect to outsourced databases such as data privacy and data confidentiality. Since PIR protocol involves matrix operations, the performance of the system is also a major issue. In order to optimize the performance of the system, analysis of the existing PIR implementation is required. The two security issues are described as:

Data privacy: Clients should not get more information than what they are querying on the server.

* Corresponding author:
sunilbmane@gmail.com (Sunil B. Mane)
Published online at http://journal.sapub.org/ijnc

Data confidentiality: Outsiders and even the server's operators (database administrators) should not be able to see the outsourced data contents in any case (including when a client's query is performed on the server). Usually it is achieved using encryption of databases.

The model of PIR scheme given in[2],[3] is a lattice based implementation of PIR. We have added encryption as an addition to existing method and analysed the order of query matrix used in PIR algorithms. A reply generation contains matrix multiplication operations it handles the data privacy issue and encryption of the result gives the data confidentiality. It has been is described in three phases as shown below:

2.1. Request Generation

The scheme has three global integer parameters: $2N$, the dimension of the lattice and special parameters p and q. The database is described as a set of n elements, and we note i_0 the index of the database element the user is interested in. To obtain a PIR request, the user will follow:

1) Note $l_0 = \lceil log\ (n \times N) \rceil + 2$ and set q as $2^{2 \times l_0 - 1}$ and p as a prime larger than $2^3 \times l_0$.

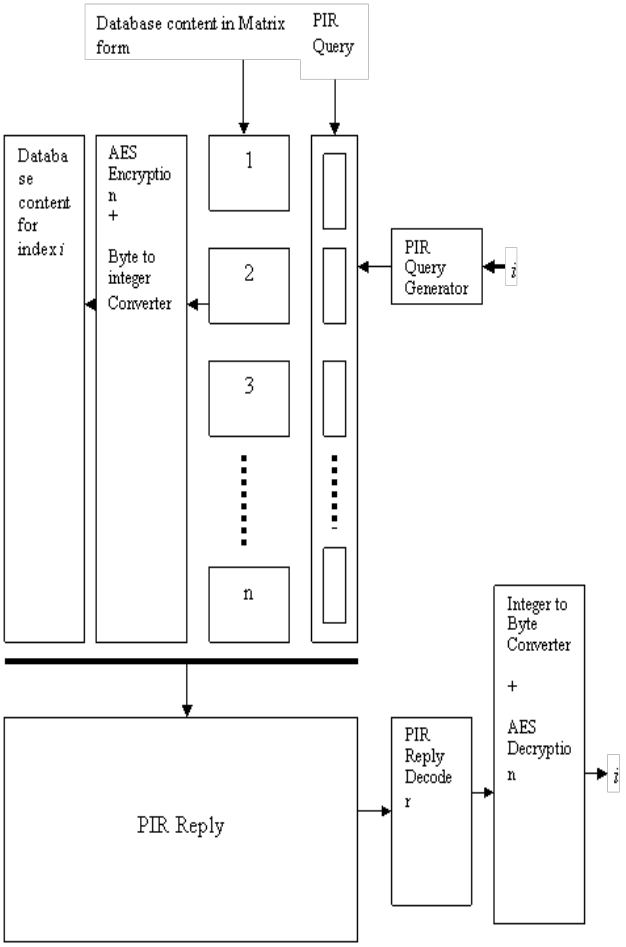


Figure 1. Proposed model of PIR Protocol

2) Generate $M1$ and $M2$, two random matrices over Z/pZ such that $M1$ is invertible, and note $M = [M1|M2]$.

3) For each $i \in \{1 \cdots N\}$ compute a matrix $B'_i = [B'_{i,1}|B'_{i,2}]$ by multiplying M to the left by a random invertible matrix Pi.

4) Generate the random scrambling matrix Δ as a $2N \times 2N$ random invertible matrix over Z/pZ.

5) For each $i \in \{1 \cdots N\} \backslash i0$ generate the soft noise matrix Di, a $N \times N$ random matrix over $\{-1, 1\}$, and compute the soft disturbed matrix $Bi = [B'i, 1|B'i, 2 + Di]\Delta$.

6) Generate Di_0, the hard noise matrix, by:

• generating a soft noise matrix,

• multiplying each diagonal term by q.

7) Compute the hard disturbed matrix $Bi0 = [B'i_0, 1|B'i_0, 2 + Di0]\ \Delta$.

8) Send the tuple $(B_1, \cdots, B_n, p)$ to the database.

2.2. Answer Encoding

Each element of the database ai (for $i \in \{1, \cdots, n\}$) is encoded as an $L \times N$ matrix Ai of l_0-bit scalars. To answer to the PIR reply the database follows.

1) Note $A = [A1| \cdots |An]$ the column concatenation of the database element matrices and $B = [B^t{}_1| \cdots |B^t{}_n]^t$ the row concatenation of the query matrices.

2) Compute $R = A \times B$ over Z/pZ.

3) Return R.

The result is a $L \times 2N$ matrix over Z/pZ. As a database element is encoded as a $L \times N$ matrix of l_0-bit scalars and p is $3l0$-bit long, the expansion factor of the reply will be $F = 6$.

2.3. Information Extraction

In order to simplify the description of the information extraction protocol we define how to operate for each row vector of the reply. In practice some of these operations are aggregated and done over many row vectors at the same time. To extract the information from a row vector Vx of the database reply, the client operates in two phases. First it unscrambles the vector and recovers the noise included in it (steps 1 and 2 of the protocol), and then he will filter out this noise to obtain the information (steps 4 and 5). Note that steps 4 and 5 are *not* done over Z/pZ, as they correspond to an Extended Euclid's division over Z.

1) Unscramble the noisy vector $V'x = Vx\Delta - 1$

2) Retrieve $E = V'x\ (D) - V'x\ (U)\ M1^{-1}M_2$,

The inserted noise, $V'x\ (U)$ and $Vx\ (D)$ being respectively the undisturbed and disturbed halves of $V'x$ (i.e. the left half and the right half respectively).

3) For each ex,y in $E = [ex,1 \cdots ex,N\ ]$, if $ex,y > p/2$ compute $e'x,y = p - ex,y$.

4) For each $e'x,y$ compute $e''x,y = e'x,y - \varepsilon$

With ε: $= e\_x,y\%q$ if $e'x,y\%q < q/2$ else

$\varepsilon := e'x,y\%q - q$.

5) For each $y \in \{1 \cdots n\}$, compute $ai_0, x, y = e''x, yq - 1$.

Query generation Algorithm creates PIR query for the input index i. The database content for the index i will be encrypted with AES encryption algorithm and its result is then converted into equivalent integers, which then will be converted into integer matrix form. The elements of this matrix are combined with the PIR query using the reply

generation algorithm. The result is then sent back to the user. Finally, the user will decode the answer through a reply decoding algorithm & AES decryption algorithm.

3. Description of the Proposed Scheme

In this section we describe an overview of our proposed PIR scheme. There are two factors which we have more focused on in this paper:

- The bottleneck in PIR schemes is reply generation and it is not reply extraction[1].
- The GPGPU (General-Purpose computation using Graphics Processing Units) computations will be on the common operations of Query generation & reply generation phase[1].

In our proposed scheme the query generation (without common operation in reply generations) and reply extraction have been implemented straightforwardly. We describe a database as a set of n records. Each record a_i is split into l_0-bits sub-elements and represented as a matrix A. N and l_0 are being two security parameters. Noting down S_{max} the size of the largest record in the database, parameter L is set to $L = \lceil S_{max} / (N \times l_0) \rceil$. If a record is smaller than $(L \times N \times l_0)$ then end of the matrix filled up with a standard padding techniques like filling all the remaining elements of the matrix as '0's.

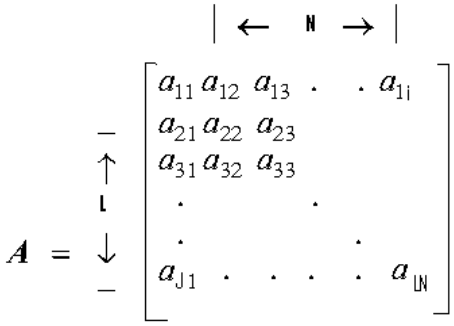


Figure 2. Database result in integer matrix form

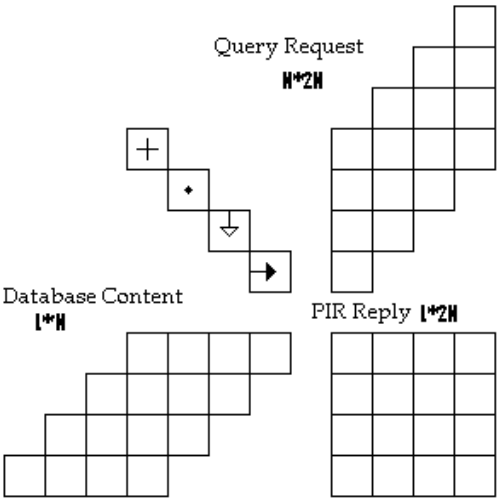


Figure 3. PIR reply generation

If a user wants to retrieve a record from the database, we are assuming that he provides the index i in order to analyse the PIR internals in detail. Then the PIR query generation algorithm creates a query formed of n matrices $B_1 \ldots B_n$, one for each record in the database. Each matrix is of order $N \times 2N$ with scalars in Z/pZ, p being a $3 \times l_0$-bit prime. If the user wants to retrieve a record a_{i0} query generation algorithm generates a query such that the i_0^{th} matrix has a special property that is invisible to the server.

This property ensures that the user will be able to extract the desired record from the server reply. The user sends the query to the server (service provider) hosting the database to generate the reply and the server multiplies the column concatenation of the database record matrices and the row concatenation of the query matrices. The resulting matrix, which we have note down as R, is a $L \times 2N$ matrix with $3l_0$-bit scalars (in Z/pZ) and thus, is six times larger than a database record matrix.

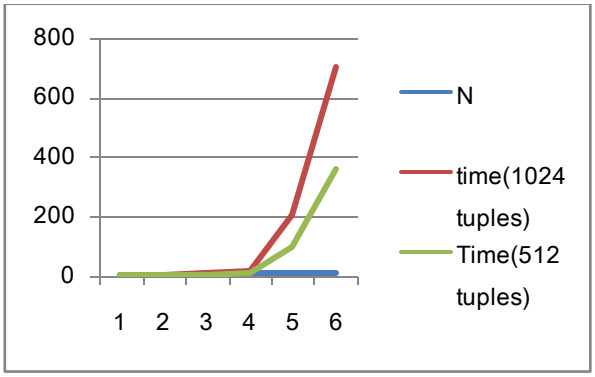
4. Implementation

We have implemented the PIR scheme in non-concurrent paradigm. As per as practical implementation is concern, it is very clear that, in this particular scenario there is a need of having parallelism. Currently we have non-concurrent version of library which containing all operation needed for PIR. We have executed all the PIR operation on following configuration:

Processor: Intel C2D T660 2.2 GHz
RAM: 4 GHz
Operating System: Ubuntu 9.10
Programming Language: Java 6

5. Experimentation and Results

Following Graph 1 plotted against Execution time *vs.* Value of dimensioning factor (N). As per the graph if dimensioning factor N increases linearly then the execution time T increases exponentially. E.g. suppose the number of records is 1024 or 512 and when the value of $N > 6$ execution time has tremendous increase as shown in the graph. Hence for high dimensioning value there is need of concurrency.



Graph 1. Time in Sec. Vs. Order of Matrix

6. Reply Generation of GPGPU

The concept of general-purpose computation using graphics processing units (or GPGPU) has arisen in recent years. GPGPU applications make use of graphic processing units (GPU) as massively parallel processors, turned away from their original purpose but nonetheless highly efficient in numerous application domains. Combining both speed and bandwidth due to their parallel architecture (and accessible cost), GPU are an attractive choice for complex computations compared to traditional CPU since they exhibit significant performance overheads, and are supported by constantly improving high-level programming language provided by both GPU vendors and the academic community.

GPU have now evolved from highly-specialized graphics processors into powerful, flexible programmable units, and become increasingly popular for a wide range of applications. All the existing PIR schemes are highly parallelizable. However, in the classic schemes, the basic operation for reply generation (per bit on the database) is a multiplication over a 1024 or 2048 bit modulus.[1] Our proposed model will parallelize all common basic operations in reply generation as well as Query matrix generation phase. The stream processors that form a GPU are not adapted to do directly such operations and trying to do a straightforward parallelization would result in very poor performance. The correct approach is to use the whole set of stream processors to split an exponentiation among them. This can be efficiently encoded through SIMD (Simple Instruction Multiple Data) instructions, on which GPUs are specialized. Each of these operations can be executed inside a single stream processor and the large number of these processors in modern GPUs results in a significant performance improvement.

7. Conclusions

Our model of Private Information Retrieval protocol gives an idea of data privacy as well as data confidentiality with the help of encryption algorithm like AES algorithm. This paper shows relation between the value of dimensioning parameter N & execution time T (in sec.) on non-concurrent implementation. This paper also propose the use of GPGPU (General-Purpose computation using Graphics Processing Units) for implementation to increase speed of execution of protocol, which will leads to practical usability of PIR schemes in real world.

REFERENCES

[1] Carlos Aguilar Melchor, “High-Speed Single-Database PIR Implementation” in 2008.

[2] Carlos Aguilar Melchor, Philippe Gaborit “A Fast Private Information Retrieval Protocol” in ISIT 2008, Toronto, Canada, July 6 - 11, 2008.

[3] Carlos Aguilar-Melchor, Philippe Gaborit “A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol” in WEWORC paper July 2007.

[4] Ian Goldberg “Improving the Robustness of Private Information Retrieval” in IEEE Symposium on Security and Privacy(SP'07) 2007.

[5] Carlos Aguilar Melchor, Philippe Gaborit “Single-Database Private Information Retrieval Protocols Overview, Usability and Trends” in march 2007

[6] Thierry P. Berger: “New perspectives for code based public key Cryptography” Darmstadt, 25 september 2006.

[7] Christian Wieschebrink, “Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography” in ISIT 2006, Seattle, USA, July 9 14, 2006.

[8] Giovanni Di Crescenzoy Tal Malkinz Rafail Ostrovskyx: “Single Database Private Information Retrieval Implies Oblivious Transfer” in EUROCRYPT 2000, LNCS 1807, pp. 122-138, 2000.

[9] Benny Chorf Oded Goldreichf Eyal Kushilevitz Madhu Sudan: “Private Information Retrieval” in 1995 IEEE.