A new multi-server scheme for private information retrieval

Chi Sing Chum and Xiaowen Zhang

Abstract. Traditional private information retrieval (PIR) schemes have been around for about two decades. They mainly consider the effectiveness of the communication complexity in the sense to minimize the number of bits transferred between the user who does the query and the server which answers the query. In this paper, we introduce a new scheme which takes both the time and communication complexities into consideration. The scheme has a simple implementation and is especially suitable if there is only a small number of replicated servers available.

1. Introduction

Private information retrieval (PIR) deals with the privacy of a user when he queries a public database. It was first introduced by Chor et al. \cite{1} in 1995. It is formalized as follows: given a database $x$ which consists of $n$ bits, $x = x_1 \ldots x_n$, a user wants to inquire the $i$th bit without letting the database know any information about $i$. A trivial solution is to let the user download the entire database. In this case, the communication complexity, which is the number of bits transferred between the user and the database, is $n$. Chor et al. \cite{1} proved that this trivial solution turned out to be optimal for a single database in the information theoretic setting. In the literature, PIR schemes were classified into information theoretic and computational. A PIR in an information theoretic setting provides perfect privacy against those databases with unlimited computational power, while computational PIR provides privacy against those databases with polynomial time computational power. This paper only considers PIR schemes under an information theoretic setting. For more information about computational PIR, please refer to \cite{3}.

The rest of the paper is organized as follow. Section 2 will introduce the PIR scheme that originally came from Chor et al. \cite{1}. In section 3 we discuss the motivations and the proposal of the new scheme. We conclude the paper in section 4.

\textbf{2010 Mathematics Subject Classification.} Primary 68P20.

\textbf{Key words and phrases.} Private information retrieval, non-colluding servers, communication complexity.
2. Private information retrieval schemes

Multi-server scheme: When the user downloads all the \( n \) bits, the database will not get any information about \( i \) even if the database has unlimited computational power. Chor et al. [1] showed that if we had more than one non-colluding servers with each having a complete database, we could reduce the communication complexity and preserve the perfect privacy as well.

Let \( k = 2^d \) be the number of databases hosted on \( k \) different non-colluding servers, and for simplicity let \( l = \sqrt[3]{n} \). Instead of considering the database as one dimensional, we think of it as \( d \) dimensional. Retrieving \( x_i \), where \( 1 \leq i \leq n \), becomes the problem of retrieving \( x_{i_1 \ldots i_d} \) where \( 1 \leq i_1, \ldots, i_d \leq l \). The user generates \( d \) random binary strings of length \( l \), i.e., \( S_1^0, S_2^0, \ldots, S_d^0 \in \{0,1\}^l \). Based on these \( d \) strings and \( i_1, \ldots, i_d \), the user constructs \( d \) binary strings \( S_1^1 \) by flipping the \( i_1 \)th bit of \( S_1^0 \) (denoted by \( S_1^1 = S_1^0 \oplus i_1 \)), \( S_2^1 \) by flipping the \( i_2 \)th bit of \( S_2^0 \) (i.e. \( S_2^1 = S_2^0 \oplus i_2 \)), \ldots, \( S_d^1 \) by flipping the \( i_d \)th bit of \( S_d^0 \) (i.e. \( S_d^1 = S_d^0 \oplus i_d \)). This way, the user gets \( d \) sets \( \{S_1^0, S_1^1\}, \ldots, \{S_d^0, S_d^1\} \), and each consists of 2 binary strings of length \( l \). We choose 1 string from each set; then we have \( 2^d \) combinations. For each combination \( \alpha = \alpha_1 \ldots \alpha_d \in \{0,1\}^d \), the user will send \( S_1^{\alpha_1}, S_2^{\alpha_2}, \ldots, S_d^{\alpha_d} \) to the database \( DB_{\alpha_1 \ldots \alpha_d} \).

The user sends out \( d \) strings of length \( l \) to each database. There are \( 2^d \) databases, so the total number of bits sent to the databases is \((dl)2^d\). Let \( j \in S \) denote the \( j \)th position in \( S \) is 1. Each database \( DB_{\alpha} \) will do the following calculation:

\[
\bigoplus_{j_1 \in S_1^\alpha, \ldots, j_d \in S_d^\alpha} x_{j_1 \ldots j_d}, 1 \leq j_1, \ldots, j_d \leq l.
\]

That means the bit \( x_{j_1 \ldots j_d} \) will be selected if \( S_1^{\alpha_1} \) contains 1 in \( j_1 \)th position, \( \ldots, S_d^{\alpha_d} \) contains 1 in \( j_d \)th position. Here only 1 bit will be sent back to the user. So, \( 2^d \) bits will be sent back.

The total communication complexity equals to

\[
2^d(dl + 1) = 2^d(dn^{1/d} + 1) = O(n^{1/d}).
\]

For any bit \( x_{j_1 \ldots j_d} \) which is different from \( x_{i_1 \ldots i_d} \), even (possibly zero) of such bits will be sent back to the user and this will cancel out each other. Only \( x_{i_1 \ldots i_d} \) will be sent back to the user one time. We will illustrate this in more detail in the following example.

**Example.** Consider that there are 8 databases (i.e., \( d = 3 \)). Suppose the user wants to retrieve \( x_{i_1, i_2, i_3} \). He first generates \( S_1^0, S_2^0, S_3^0 \), and then constructs

\[
S_1^1 = S_1^0 \oplus i_1, \quad S_2^1 = S_2^0 \oplus i_2, \quad S_3^1 = S_3^0 \oplus i_3.
\]

He then sends out

\[
S_1^0, S_2^0, S_3^0 \text{ to } DB_{000}, \quad S_1^0, S_2^0, S_3^1 \text{ to } DB_{001}, \quad \ldots, \quad S_1^1, S_2^1, S_3^1 \text{ to } DB_{111}.
\]

3 binary strings of length \( l \) will be sent to each database. Altogether \( 8 \times 3 \times l \) bits will be sent. Each database sends back 1 bit, so 8 bits will be received by the user.

The total number of bits exchanged is \( 8 \times 3 \times l + 8 = 8(3l + 1) = 8(3n^{1/3} + 1) \).

Therefore the communication complexity is \( O(n^{1/3}) \).

Consider any bit \( x_{j_1, j_2, j_3} \), and let \( S(j) \) be the \( j \)th bit of the string \( S \).
(i) If \( j_1 \neq i_1, j_2 \neq i_2 \) and \( j_3 \neq i_3 \), then \( S^0_1(j_1) = S^1_1(j_1), S^0_2(j_2) = S^1_2(j_2), S^0_3(j_3) = S^1_3(j_3) \). Either 0 or 8 of \( x_{j_1,j_2,j_3} \) will be sent back.

(ii) If two of \( j \)s are different from the corresponding \( i \), say \( j_1 \neq i_1, j_2 \neq i_2 \), then \( S^0_1(j_1), S^0_2(j_2) = S^1_2(j_2), S^0_3(j_3) \neq S^3_3(j_3) \). Either 0 or 4 of \( x_{j_1,j_2,j_3} \) will be sent back.

(iii) If one of \( j \)s is different from the corresponding \( i \), say \( j_1 \neq i_1 \), then \( S^0_1(j_1) = S^1_1(j_1), S^0_2(j_2) \neq S^1_2(j_2), S^0_3(j_3) \neq S^3_3(j_3) \). Either 0 or 2 of \( x_{j_1,j_2,j_3} \) will be sent back.

(iv) If \( j_1 = i_1, j_2 = i_2, j_3 = i_3 \), then \( S^0_1(j_1) \neq S^1_1(j_1), S^0_2(j_2) \neq S^1_2(j_2), S^0_3(j_3) \neq S^3_3(j_3) \). In this case only 1 of \( x_{j_1,j_2,j_3} \) will be sent back.

That means when the user XORs all the 8 bits from the 8 databases, he can get the desired bit \( x_{i_1,i_2,i_3} \).

**The covering code scheme:** Chor et al. [1] has a detailed discussion about the covering code scheme. Briefly, a \( DB_\alpha \) can simulate any other \( DB_\alpha' \) if the hamming distance between \( \alpha \) and \( \alpha' \) is 1. It begins with \( 2^d \) databases and the communication complexity \( O(n^{1/d}) \). The number of bits now sent back from \( DB_\alpha \) will be changed from 1 to \( O(n^{1/d}) \). Since both the user and the databases send \( O(n^{1/d}) \) bits, the communication complexity remains \( O(n^{1/d}) \) but the number of databases can be reduced.

Suppose we have 2 databases \( DB_000 \) and \( DB_{111} \). \( DB_000 \) works as before so it sends back 1 bit. In addition, it takes over the function of \( DB_{100} \). \( DB_{000} \) receives \( S^0_1, S^2_2, S^3_3 \), while \( DB_{100} \) receives \( S^1_1, S^0_2, S^0_3 \). \( DB_000 \) does not know what \( i_1 \) is, but it does know that there is only 1 bit difference between \( S^0_1 \) and \( S^1_1 \) at the \( i_1 \)th bit, i.e., \( S^0_1(i_1) \neq S^1_1(i_1) \). \( DB_000 \) can simulate \( DB_{100} \) by flipping one bit at a time from \( S^0_1 \) until all \( l \) bits were flipped. It will send back \( l \) bits. The user knows what \( i_1 \) is, so he can pick up the right bit and ignore the others. \( DB_000 \) can simulate \( DB_{010} \) and \( DB_{001} \) in the same manner. Same idea applies to \( DB_{111} \). It can simulate \( DB_{011}, DB_{101}, \) and \( DB_{110} \). All together \( 6l + 2 \) bits will be sent back. Since the user sends out \( 6l \) bits, the total bits exchanged is \( 12l + 2 = 12n^{1/3} + 2 \). Thus the communication complexity is \( O(n^{1/3}) \).

This technique shows how to reduce the number of databases while keeping the same communication complexity. In this case, we can reduce from 8 to 2 databases while keeping the communication complexity to \( O(n^{1/3}) \). In terms of communication complexity, this scheme achieves the optimal performance for 2 servers.

### 3. A new PIR scheme

Since the original paper by Chor et al. [1], there are a lot of improvements for the communication complexity. Please refer to [2] [4] for a detailed survey. However, it is our intention to look for a scheme which balances the communication complexity and computing efficiency, especially under a practical setting namely only a small number of servers, say 2 (another one as a backup) to 4, are available. Another reason is that we want to have a practical scheme which is easy to implement. Based on the above scheme, we use the same idea of adjusting the communication complexity of the user and servers to come up with the following new scheme. We divided a \( d \)-dimensional \( DB \) into many smaller \( d \)-dimensional \( dbs \). So the size of the random strings sent to the servers can be cut down as the databases \( dbs \) are smaller. On the other hand, the number of bits sent back from
the servers will be increased due to smaller databases \(db\) s. If there is a scheme that the communication complexity from the user side is much greater than that from the servers side, we can use this approach to make a balance and hence to reduce the overall communication complexity.

**Proposed new scheme:** In the multi-server scheme, we consider the \(DB\) as \(d\)-dimensional, i.e., \(n = \frac{1}{d} \times \ldots \times \frac{n}{d}\). And let \(l = \frac{n}{d}\). We now divide each dimension into \(\frac{n}{d(d+1)}\) equal parts. Each part has length \(l' = \frac{n}{d(d+1)} = \frac{n}{d+1}\). The original \(DB\) is now divided into smaller \(d\)-dimensional \(db\) s of equal size which equals \(\frac{n}{d+1} \times \ldots \times \frac{n}{d+1} = \frac{n}{d+1}\). The number of such \(db\) s is \(n = \frac{n}{d+1}\).

Suppose we want to retrieve \(x_{i_1,\ldots,i_d}\), \(1 \leq i_1,\ldots,i_d \leq l\), in the original \(DB\). Now this bit will be in one of the \(db\) s and let the position be \(x_{i'_1,\ldots,i'_d}\), \(1 \leq i'_1,\ldots,i'_d \leq l'\). We apply the original multi-server scheme to retrieve the bits at position \(x_{i'_1,\ldots,i'_d}\) from all the \(db\) s. We just keep the desired bit and ignore those bits that we do not need. Since we apply the same random inquiry string to all \(db\) s, the communication complexity will be \(O(n^{1/d+1})\). There are \(n^{1/d+1}\) \(db\) s, so the communication complexity from the servers will also be equal to \(O(n^{1/d+1})\). Hence the total communication complexity is \(O(n^{1/d+1}) + O(n^{1/d+1}) = O(n^{1/d+1})\).

By breaking down the original \(d\)-dimensional \(DB\) into smaller \(d\)-dimensional \(db\) s of equal size, the new scheme reduces the communication complexity from \(O(n^{1/d})\) to \(O(n^{1/d+1})\).

**Example 1.** Let \(d = 1\), that means there are 2 servers. In the new scheme, the original \(DB\) of length \(n\) (up-side) in Fig. 1 will be divided into \(\sqrt{n}\) \(db\) s each with size of \(\sqrt{n}\), as shown in (low-side) Fig. 1. Suppose we want to retrieve \(i\) th bit \((1 \leq i \leq n)\) from the original \(DB\). Now this bit lies within \(db_j\) \((1 \leq j \leq \sqrt{n})\) at position \(i\) th, \(1 \leq i \leq \sqrt{n}\) as shown in Fig. 1. Retrieving \(i\) th bit from the original \(DB\) becomes getting \(i\) th bit from all \(db\) s. The user will ignore all the bits returned from the servers with the exception of the one from \(db_j\). The communication complexity for both the user and the servers is equal to \(O(n^{1/2})\). Therefore the overall communication complexity is also equal to \(O(n^{1/2})\).

**Example 2.** Let \(d = 2\), i.e., there are 4 servers. In the new scheme, the original \(DB\), viewed as a square of \(\sqrt{n} \times \sqrt{n}\) (left-side) in Fig. 2 will now be divided into \(n^{1/3}\) \(db\) s (squares) of size \(n^{2/3}\), as shown (right-side) in Fig. 2.

Suppose the bit that we want to retrieve \((i,j)\) \((1 \leq i,j \leq \sqrt{n})\) (left-side) now lies within \(db_{l,k}\) \((1 \leq l,k \leq n^{1/3})\) with position \((i',j')\) \((1 \leq i',j' \leq n^{1/3})\). In the new scheme, all the \((i',j')\) bits of all \(db\) s will be returned to the user, but only the \((i',j')\) bit within \(DB_{l,k}\) will be used and the others will be ignored. Since both the length of the \(db\) s and the number of small \(db\) s equal \(n^{1/3}\), the overall communication complexity equals \(O(n^{1/3})\).

**Remarks.** We need to seek a point that balances the two complexities. That means both the user and server complexities are the same. After the balance point,
if we keep on reducing the size of the random strings, then the server complexity will be greater than that of the user, and hence the overall communication complexity will be increased. Recall in the Example 1, if we divide the original $DB$ into $dbs$ of length $n^{1/3}$, instead of $n^{1/2}$, the number of $dbs$ will be equal to $n^{2/3}$. Then the overall communication complexity will be $O(n^{1/3}) + O(n^{2/3}) = O(n^{2/3})$.

4. Conclusions

Based on Chor et al. [1], we come up with a new scheme which reduces the communication complexity from $O(n^{1/d})$ to $O(n^{1/d+1})$ when $2^d$ servers are available. This does not get too much noticeable advantages when $d$ is large and in fact many existing schemes provide much better performances. However, if only a small number of servers are available, then this new scheme will provide a good balance between communication and computational efficiencies together with a simple implementation. For further research we would like to look for the possibility of applying this new technique to any existing scheme for improvements.

Acknowledgments

Authors would like to thank the referees for their valuable suggestions and comments. This work is supported, in part, by a PSC-CUNY Research Award.
References


Computer Science Dept., Graduate Center, CUNY, New York, New York
E-mail address: CChum@gradcenter.cuny.edu

Computer Science Dept., College of Staten Island, CUNY, Staten Island, New York. Computer Science Dept., Graduate Center, CUNY, New York, New York
E-mail address: Xiaowen.Zhang@csi.cuny.edu