CMSC 858K — Advanced Topics in Cryptography	April 13, 2004

Lecture 20

Lecturer: Bill Gasarch *Scribe(s): Rengarajan Aravamudhan, Julie Staub, Nan Wang*

1 Last Lecture Summary

In lecture 18, we showed a technique that can be used to achieve k-database PIR with communication complexity $\mathcal{O}\left(n^{1/(\log k+\log\log k)}\right)$ (we did not prove this bound in class, but the method we showed yields this bound). Today, we show an improved approach due to Ambainis [1] that achieves k-database PIR with $\mathcal{O}(n^{1/2k-1})$ communication complexity.

2 A 3-Database PIR Protocol with $\mathcal{O}(n^{1/5})$ Communication

We start by illustrating the technique for the case of 3 databases, using the notation as in lecture 18 (in fact, this scheme builds on the previous schemes, and we assume the reader is familiar with lecture 18). Here, we model the n-bit database as an $n^{1/5}\times n^{1/5}\times n^{1/5}\times n^{1/5}\times n^{1/5}$ table $\{x_{i_1,i_2,i_3,i_4,i_5}\}$, with the desired bit indexed as $(i_1^*,i_2^*,i_3^*,i_4^*,i_5^*)$. The three databases are DB_1, DB_2, and DB_3. We first discuss the intuition, and then describe the actual protocol.

As in the previous schemes we have seen, the user begins by choosing five random sets $S_1^0,S_2^0,S_3^0,S_4^0,S_5^0\subseteq\{1,2,...,n^{1/5}\}$, and computing $S_1^1=S_1^0\oplus\{i_1^*\}$, $\ldots$, $S_5^1=S_5\oplus\{i_5^*\}$. The user will send $S_1^0,S_2^0,S_3^0,S_4^0,S_5^0$ to DB_1 and DB_2, and send $S_1^1,S_2^1,S_3^1,S_4^1,S_5^1$ to DB_3. Let

$$X_{b_1b_2b_3b_4b_5}\stackrel{\text{def}}{=}\bigoplus_{i_1\in S_1^{b_1}}\bigoplus_{i_2\in S_2^{b_2}}\bigoplus_{i_3\in S_3^{b_3}}\bigoplus_{i_4\in S_4^{b_4}}\bigoplus_{i_5\in S_5^{b_5}}x_{i_1,i_2,i_3,i_4,i_5}.$$

Recall (from the schemes we have seen in lecture 18) that the user would like to obtain all 32 of the values $X_{00000},\ldots,X_{11111}$, and then xor these together to recover the desired bit. Now, DB_1 (and DB_2) can easily compute X_{00000}, while DB_3 can easily compute X_{11111}. Furthermore, since the user is already sending $O(n^{1/5})$ communication to the databases we may as well let the databases send this much communication back. We saw (in the improved scheme from lecture 18) that this can help because we may then have, for example, DB_3 compute X_{11110} *for all* $n^{1/5}$ *possible values of* S_5^0, and then send these $n^{1/5}$ bits back to the user (who selects and uses the one he is interested in). Adopting this approach, we see that DB_3 can send back (enough information for the user to compute) $X_{11110},X_{11101},X_{11011},X_{10111}$, and X_{01111}. Similarly, either of DB_1 or DB_2 can send back (enough information for the user to compute) $X_{00001},X_{00010},X_{00100},X_{01000}$, and X_{10000}. This can be all done while maintaining communication complexity $\mathcal{O}(n^{1/5})$.

Unfortunately, we are not yet done because the user will still be missing 20 of the values he needs to recover the desired bit. Note that we cannot extend the above approach in the trivial way to allow the user to recover the necessary values without exceeding $\mathcal{O}(n^{1/5})$ communication complexity: if we have DB_1 send back X_{11000} for all possible values of S_1^1 *and* S_2^1, this will require DB_1 to send $n^{2/5}$ bits.

Instead, what we do is to apply a 2-database PIR protocol as a subroutine. Namely, *both* DB_1 and DB_2 will compute each of the remaining values for all possibilities of each of the unknown sets. (For example, DB_1 and DB_2 will compute X_{11100} for all possible values of S_1^1, S_2^1, and S_3^1.) This results in each of these databases holding the same copy of 20 strings, each of length at most $n^{3/5}$. The key point is that *the user only needs one bit from each of these strings* (i.e., the bit corresponding to the actual value of $S_1^1, \ldots, S_5^1$) and this bit is known by the user in step 1. Since DB_1 and DB_2 hold identical copies of these strings, they and the user can use multiple invocations of a 2-database PIR protocol with communication complexity $\mathcal{O}(N^{1/3})$ (we saw such a scheme last time) to allow the user to obtain the desired bits from each of these strings. Since $N \leq n^{3/5}$ in our case, this will require communication complexity $\mathcal{O}(n^{1/5})$ meaning that the overall communication complexity of the entire protocol remains $\mathcal{O}(n^{1/5})$.

Using this intuition, we obtain the following protocol:

1. The user begins by choosing five random sets $S_1^0, S_2^0, S_3^0, S_4^0, S_5^0 \subseteq \{1, 2, ..., n^{1/5}\}$, and computing $S_1^1 = S_1^0 \oplus \{i_1^*\}, \ldots, S_5^1 = S_5 \oplus \{i_5^*\}$. The user sends $S_1^0, S_2^0, S_3^0, S_4^0, S_5^0$ to DB_1 and DB_2, and sends $S_1^1, S_2^1, S_3^1, S_4^1, S_5^1$ to DB_3.

2. The user also sends to DB_1 and DB_2 20 queries for any 2-database PIR protocol with $\mathcal{O}(n^{1/3})$ comm. complexity. These queries are determined based on the sets that the user generated in the previous step.

3. DB_1 and DB_3 send back X_{00000} and X_{11111}, respectively. Also, DB_1 sends back $X_{00001}, \ldots, X_{10000}$ for all $n^{1/5}$ possible values for each of $S_1^1, \ldots, S_5^1$. Similarly, DB_3 sends $X_{11110}, \ldots, X_{01111}$ for all $n^{1/5}$ possible values for each of $S_1^0, \ldots, S_5^0$.

4. (We use X_{11100} as an example, but exactly the same computation is carried out for each of the remaining values.) DB_1 and DB_2 both generate X_{11100} for all $n^{3/5}$ possibilities of S_1^1, S_2^1, S_3^1. This results in each of these databases holding identical copies of a string of length $n^{3/5}$. Using the appropriate query that was sent by the user, each database computes a response using the underlying 2-database PIR protocol.

5. The user obtains X_{00000} and X_{11111} immediately, and can easily select the values for $X_{00001}, \ldots, X_{10000}$ and $X_{11110}, \ldots, X_{01111}$ from the data sent back by the databases. For the remaining values, the user runs the underlying PIR protocol using the appropriate replies sent back by DB_1 and DB_2 to recover all remaining values. The desired bit of the original data is recovered as:

$$\bigoplus_{b_1=0}^{1} \bigoplus_{b_2=0}^{1} \bigoplus_{b_3=0}^{1} \bigoplus_{b_4=0}^{1} \bigoplus_{b_5=0}^{1} X_{b_1 b_2 b_3 b_4 b_5}.$$

3 Extending the Scheme for k Databases

We generalize the scheme of the previous section and prove the following theorem:

Theorem 1 *For all $k \geq 2$, there exists a k-database PIR scheme with $\mathcal{O}(n^{1/2k-1})$ communication complexity.*

Proof We will prove this by induction. For $k = 2, 3$, we have already shown that the theorem holds. So, assume the theorem is true for $k-1$, and there exists a $(k-1)$-database PIR scheme with $\mathcal{O}(n^{1/2k-3})$ communication complexity. We show how to construct a k-database scheme as claimed by the theorem.

We view the n-bit database as an $n^{1/2k-1} \times n^{1/2k-1} \times \cdots \times n^{1/2k-1}$ array with the desired bit indexed as $(i_1^*, i_2^*, \ldots, i_{2k-1}^*)$. We use the notation from the previous section. The protocol is defined as follows:

1. The user chooses $2k-1$ random sets $S_1^0, \ldots, S_{2k-1}^0 \subseteq \{1, \ldots, n^{1/(2k-1)}\}$, and computes $S_\ell^1 = S_\ell^0 \oplus \{i_\ell^*\}$. The user sends $S_1^0, \ldots, S_{2k-1}^0$ to each of $DB_1, \ldots, DB_{k-1}$ and sends $S_1^1, \ldots, S_{2k-1}^1$ to DB_k.

2. DB_1 will compute and send $X_{0^{2k-1}}$ as well as the $n^{1/(2k-1)}$ possibilities for each of $\{X_{0^i 1 0^{2k-2-i}}\}_{i=0}^{2k-2}$. Similarly, DB_k will compute and send $X_{1^{2k-1}}$ as well as the $n^{1/(2k-1)}$ possibilities for each of $\{X_{1^i 0 1^{2k-2-i}}\}_{i=0}^{2k-2}$.

3. In addition, each of the databases $DB_1, \ldots, DB_{k-1}$ will compute all possible values for X_w for all $(2k-1)$-bit strings w having at least 2 and at most $2k-3$ ones. Instead of sending these directly to the user, the databases will execute the $(k-1)$-database PIR protocol (that exists by assumption) on these strings, using queries sent by the user in the first stage.

The total communication complexity can be computed as follows:

- Sending the sets from the user to all k databases requires $k \cdot (2k-1) \cdot n^{1/(2k-1)}$ bits.

- Steps 2 and 3 each require only $(2k-2) \cdot n^{1/(2k-1)}$ bits to be sent by databases DB_1 and DB_k.

- Step 4 involves running a $(k-1)$-database PIR protocol on fewer than 2^{2k-1} strings, each of length at most $n^{(2k-3)/(2k-1)}$. Since the underlying PIR protocol has communication complexity $\mathcal{O}(N^{1/(2k-3)})$, this requires total communication (including the communication from the user in the first round) $\mathcal{O}(n^{1/(2k-1)})$.

The total communication complexity is therefore $\mathcal{O}(n^{1/(2k-1)})$, as desired. ■

When including the dependence on k, the communication complexity is $\mathcal{O}(2^{k^2} n^{1/(2k-1)})$. This is quite high even for moderate values of k! However, k-database PIR schemes with communication complexity $\mathcal{O}(k^3 n^{1/2k-1})$ and $\mathcal{O}(n^{(\log\log k)/(k \log k)})$ (ignoring the constant which depends on k) are known. See http://www.cs.umd.edu/~gasarch/pir.

References

[1] A. Ambainis. Upper Bound on the Communication Complexity of Private Information Retrieval. ICALP, 1997.