

Constant-Round Private Database Queries

Nenad Dedic* and Payman Mohassel **

Abstract. We consider several private database query problems. The starting point of this work is the *element rank* problem: the server holds a database of n integers, and the user an integer q ; the user wishes to find out how many database records are smaller than q , without revealing q ; nothing else about the database should be disclosed. We show a non-interactive communication-efficient solution to this problem. We then use it to solve more complex private database queries: range queries, range queries in plane and higher-dimensional generalizations of element rank. We also show an improved solution to the k^{th} ranked element problem [1], and a solution to *private keyword search* [11] using weaker assumptions than those of [11]. All our solutions assume semi-honest adversarial behaviour.

1 Introduction

Private information retrieval (PIR) allows a user to query the server’s database, without revealing information about the query. If the user cannot find out more information than required, then we have *symmetrically private information retrieval* (SPIR). Efficient solutions currently exist for limited types of queries: retrieval *by address* (the first considered PIR problem [8, 16]), retrieval *by keyword* [11]. More complex queries can be processed as well, but less efficiently [7, 19] or using relatively strong computational assumptions [15]. It is therefore desirable to find efficient solutions to more specific private database query problems.

For example, a new bidder in an auction might be allowed to learn how his bid ranks compared to other bids, but unwilling to disclose the bid to the auction server. Or consider a user who wants to compare his score against the database of an online testing service provider, while keeping his score private. These are examples of the *element rank problem*. As another motivating example, consider a driver who wishes to get a list of all fast-food joints in a city block, but does not want to disclose his location. This is an example of *2-d range retrieval*.

In this work we mainly consider the (generalization of) element rank problem. Based on this we show solutions to several other problems (e.g. range retrieval, 2-d range retrieval, shared element rank). We seek to minimize two important efficiency measures: *communication complexity* and *round complexity*. Minimal round complexity is noteworthy because it allows for easier integration with other protocols; we exemplify this by using the non-interactive nature of our basic rank protocol to solve other problems. Note that by *non-interactive*, we mean protocol with *one round* of communication between the parties.

* Department of Computer Science, Boston University. nenad@cs.bu.edu. Part of this work was done while the author was visiting the IPAM institute at UCLA.

** Department of Computer Science, University of California Davis. mohassel@cs.ucdavis.edu. Part of this work was done while the author was visiting the IPAM institute at UCLA. This work was partially supported by the Packard Foundation.

1.1 Contributions

We solve the following private database query problems, assuming semi-honest behaviour.

Interval labeling problem. As the main building block, we design a non-interactive low-communication solution to the following problem. Two players are involved: the user whose input is an integer q , and the server whose input is a sorted sequence of integers x_1, \dots, x_n , defining $n + 1$ intervals, labeled with strings v_0, \dots, v_n . The user wishes to find out the label of the interval to which q belongs. Our most efficient solution can be based on homomorphic encryption and PIR [16].

Element rank; generalizations. We then show how to use interval labeling to efficiently solve the rank problem, and generalizations of interval labeling (for example, *rectangle labeling* where the server subdivides the plane in rectangles and labels each rectangle).

Equivalence of PIR and PKS. On a more theoretical note we show that, somewhat surprisingly, non-interactive PIR alone suffices to implement non-interactive PKS. Communication complexity is (in asymptotic terms) almost the same as that of the underlying PIR. Previous solutions known to us use more specific algebraic assumptions (e.g. [11] requires homomorphic encryption). However, in a practical scenario, the solution of [11] would likely be preferable because of greater efficiency. We note that as a corollary we obtain that the interval labeling problem and its generalizations can be efficiently implemented using PIR alone.

Secure computation of k^{th} -ranked element. In [1], Aggrawal et al. design a protocol for secure computation of the k^{th} -ranked element (e.g. the median) in the union of private databases held by two parties. We use our new non-interactive protocol for secure computation of rank of an element in a database as a subprotocol, and design more efficient protocols for this task. More specifically, we design a new protocol with $\log \log(k)$ rounds of interaction between the two parties, as opposed to $\log(k)$ rounds of interaction required by the protocol of [1].

Constant-round range queries; on a line or in a plane. The server holds a database of points on a line (or, for example, on a plane). The user holds a high point and a low point on that line (or an axis-aligned rectangle in the plane). As the output of the protocol the user should only learn the points in the database that belong to the range defined by his input values, i.e., all the points in the database that lie between the high and low points (or all the points that lie inside user's rectangle). We design constant-round protocols for this problem, with communication linear in the number of retrieved points. Our protocols provide privacy for both user and the server.

1.2 Related Work

Private information retrieval (PIR), introduced by Chor, et al. in [8] is the prototypical problem of private database query evaluation. It requires that the user with input i learn

i -th entry of some database D . Security requirement is that i remain hidden to everyone but the user. Another, essential, requirement is that the total communication be strictly less than the size of D (else the problem is trivially solved by sending D). The solution given in [8] mandates statistical security, and requires multiple non-communicating servers holding copies of D . The multiple-server requirement was lifted in [16], at the necessary expense of downgrading to computational security. [12] introduce *symmetric* PIR (SPIR), where server privacy is required: the user must learn nothing beyond $D[i]$. SPIR can be seen as communication-efficient oblivious transfer [22]. The works of [20, 2] show how to convert any PIR to SPIR, at little cost.

Many solutions to the PIR problem have appeared since [5, 6, 18], improving the efficiency, or weakening/changing the underlying hardness assumptions. Most of these solutions share an attractive feature – they are *non-interactive* in the sense that they use only a single communication round (clearly this is the minimal possible number of rounds in order to achieve sublinear communication, since to learn anything at all about D , the user must at least send his query and receive the answer). This non-interactivity is noteworthy, because it allows us to look at PIR in the framework of public-key program obfuscation [21]: user’s query can be seen as an obfuscated program which the server executes on his database. Another benefit of non-interactivity lies in easier integration with more complex protocols [11]. Finally non-interactivity seems a crucial property that allows particularly efficient solutions to PIR [16, 6, 18] – they all follow this basic pattern: a non-interactive scheme with relatively high communication complexity is given first, and it is then recursively applied to obtain an efficient (still non-interactive) scheme; this recursive application is possible thanks to non-interactivity.

General multi-party computation (e.g. see [14] and [13]) can be used to solve private database query problems, but not very efficiently. It is therefore of interest to seek more efficient solutions, even if they are more specialized. We focus specifically on minimizing communication and interaction. *Communication-efficient secure function evaluation* [19] gives a protocol for secure computation of circuits with lookup tables. Such a protocol takes any oblivious Turing machine with RAM access and turns it into a private two-party protocol. While this can be used to implement a communication-efficient element rank protocol, the construction is not concerned with optimizing number of rounds of *interaction* between the parties. *Computing branching programs on encrypted data* is introduced in an independent work [15]. This work considers queries which can be represented as branching programs and gives a non-interactive solution to this problem. The construction given in [15] is more general than our private element rank protocol and can be used to implement such a task. But, we use different techniques, and by doing so can depend on more *general* and *weaker* assumptions. Furthermore, some of the protocols we study in this paper such as private range queries (see section 4) cannot be captured efficiently using branching programs.

Private keyword search (PKS) [11, 7] is a natural extension of SPIR where the server holds a set of key-value pairs (k_i, v_i) , and the user wishes to find a value associated to some keyword k if it exists. [7] in fact show how to privately query any server-held data structure, using multiple interaction rounds. *Private keyword search on streaming data* [21] considers a different model, where tagged documents come in streaming, and an untrusted computer keeps, in an encrypted form, only those which have certain

user-specified combination of tags. This combination remains hidden to the untrusted computer. *Oblivious binary search and secure range queries* [10] deals with some extensions to PIR (e.g. retrieving entries at addresses $i, i + 1, \dots, j$), as well as with computing element rank, but using a logarithmic number of rounds; this is then applied in a range retrieval protocol, where the user whose input is (a, b) finds out all database entries with *values* between a and b .

A closely related problem of non-interactive secure computation is considered in [4]. Their setting involves a public circuit C , Alice with private input x and Bob with private input y . After a single round of interaction Alice should learn $C(x, y)$ and Bob should learn nothing. They solve this problem for any polynomial-size circuit C . Communication complexity, however, depends polynomially on the sizes of x, y , even if the size of $C(x, y)$ is very small.

2 Preliminaries

Notation and conventions. We interpret binary strings as integers when convenient, with the most significant bit to the left. $[N]$ denotes the set $\{0, 1, 2, \dots, N\}$, and $[N)$ denotes $[N - 1]$. $|x|$ denotes the length of x . ϵ denotes the empty string. We write $x||y$ to denote the concatenation of strings x and y . We write x^- to denote x with the last symbol dropped (e.g. $aaba^- = aab$), and for a binary string x we write x' to denote x with the last bit flipped (e.g. $1100' = 1101$). We write $p \preceq s$ to denote that p is a prefix of s ; and $p \prec s$ to denote $p \preceq s \wedge p \neq s$.

2.1 Oblivious and Succinct Computation

In this section we introduce *oblivious computation*, a useful abstraction for certain non-interactive cryptographic tasks. This notion appears implicitly in [23] and more explicitly in [4]. We systematize and make explicit its important features. Intuitively, oblivious computation involves two parties: the *user Alice* holding her input x , and the *server Bob* holding his input y . Some function f is given publicly, and Alice wishes to learn $f(x, y)$. Bob should learn nothing, and Alice should learn nothing beyond $f(x, y)$. Alice will send some query $ex = Q(x)$ (which hides x) to Bob, who will produce an answer $a = A(ex, y)$ and send it back to Alice. Alice will then decode this answer to get $D(a) = f(x, y)$. Formally:

Definition 1. We say that a quadruple of PPT algorithms (G, Q, A, D) obviously computes a function $f(x, y)$ if for (PK, SK) generated by $G(1^k)$:

Correctness holds: $D(SK, A(PK, Q(SK, x), y)) = f(x, y)$.

Query secrecy holds: for any x, x' , the distributions $Q(SK, x)$ and $Q(SK, x')$ are computationally indistinguishable

Answer security holds: some PPT algorithm S simulates $A(PK, Q(SK, x), y)$ given only $(PK, x, f(x, y))$.

G is called the instance generator, Q the query computer, A the answer computer and D the answer decoder.

This definition naturally corresponds to one-round secure two-party computation, where only Alice gets the output. Any polynomial-sized function f can be obviously computed using one-round oblivious transfer and Yao’s garbled circuits [4]. But we will in fact consider a more stringent setting, where no general solution is known. The server holds a large database $X = (x_1, \dots, x_n)$ and the user holds a short query q . They are interested in obviously computing some function $f(q, X)$ whose output size is independent of n , while minimizing communication. Formally:

Definition 2. We say that a quadruple of PPT algorithms (G, Q, A, D) succinctly computes $f(q, X = (x_1, \dots, x_n))$ if:

Obliviousness holds: (G, Q, A, D) obviously computes f

Succinctness holds: there is a polynomial p such that

$$|Q(\text{SK}, q)| + |A(\text{PK}, Q(\text{SK}, q), X)| \leq p(\log n) \text{ for sufficiently large } n.$$

Succinct computation is a natural umbrella definition for non-interactive cryptographic tasks with a communication restriction. Some examples, which we use in this work, are:

Private information retrieval is succinct computation of the function

$$f(i, (x_1, \dots, x_n)) = x_i.$$

Solutions are given, for example, in [6, 18]. Note that most definitions of PIR impose no limit on the number of rounds. But almost all PIR protocols from the literature are in fact non-interactive (recall, this means that they require a single round), and therefore conform to the definition of succinct computation. This can be used to good effect in protocols where PIR is a building block, so for purposes of this work we take PIR to be non-interactive.

Private keyword search [11] is succinct computation of the function

$$PKS(w, (x_1, \dots, x_n, v_1, \dots, v_n)) = \begin{cases} v_a, & \exists a \ w = x_a \\ \perp, & \text{otherwise.} \end{cases}$$

We also use *contiguous PIR* – a natural generalization of PIR, where a contiguous block of s records is retrieved, while communicating $\Theta(s)$ bits. Note that the server must necessarily learn s , simply by counting the number of bits sent. The corresponding functionality is

$$ContPIR((i, s); (x_1, \dots, x_n)) = ((x_i, \dots, x_{i+s-1}); s).$$

[10] give a solution to this problem.

3 Computing Element Rank and Related Functions

We consider the following problem, which is at the heart of all the protocols in this paper.

Problem 1 (Interval Labeling). Two parties, the user Alice and the server Bob are involved. Let $L = 2^l - 1$, n and m be public integer parameters. **Parties' inputs** are as follows:

Server holds $x_1, \dots, x_n \in [L]$ and $v_0, \dots, v_n \in \{0, 1\}^m$. x_1, \dots, x_n is an increasing sequence of numbers subdividing the interval $[L]$ into $n + 1$ disjoint sub-intervals $(x_0, x_1], (x_1, x_2], \dots, (x_n, x_{n+1}]$ (where we take by convention $x_0 = -\infty$, $x_{n+1} = +\infty$). In each interval $(x_i, x_{i+1}]$ the server writes a string label v_i .

User holds an integer $q \in [L]$.

The output which the user wishes to find out is the label v_i of the interval $(x_i, x_{i+1}]$ to which q belongs. **Security** requirements are: 1. Server learns nothing, 2. User learns nothing except v_i . **Efficiency** requirements are: 1. computations must be polynomial time, 2. communication complexity must be $\text{polylog}(nl)$.

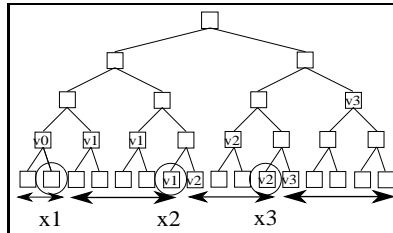
3.1 Succinctly Computing Interval Labeling and Element Rank

Let us see how to reduce succinct computation of interval labeling to private keyword search (PKS). It will be shown then that this implies a solution to element rank. An obvious way, but too expensive one, would be to create a PKS database of L entries: for every $y \in [L]$ insert (y, v_i) into the database where i is such that $y \in (x_{i-1}, x_i]$. Searching for the keyword q in this database would clearly return the correct answer v_i . We now show how the database size can be reduced from L to $2ln$, by inserting only certain prefixes of x_i 's.

We refer the reader to Section 2 to review the notation and conventions. In this section we use additional notation: for a binary string $s \in \{0, 1\}^j$, let \bar{s} denote the interval of integers $[s||0^{l-j}, s||1^{l-j}]$. Also, for user's query q , let q_i denote its i -th bit.

Database construction. The main observation of this section is the following. Consider $w \in \{0, 1\}^m$ and i such that $\bar{w} \subseteq (x_{i-1}, x_i]$ and $\bar{w}' \not\subseteq (x_{i-1}, x_i]$. Then (w, v_i) can be added to the PKS database without introducing any ambiguity. Indeed, all integers beginning with the prefix w , fall into the same interval $(x_{i-1}, x_i]$, and they therefore share the label v_i . Moreover, for any predecessor $u \prec w$, all integers beginning with u span more than one interval.

This is illustrated in the following picture. Leaves of the tree represent the integers of $[L]$, with 0 in the left, and increasing towards right. Database values x_i are circled, and intervals $(x_{i-1}, x_i]$ are indicated by arrows. Nodes corresponding to w 's described above are labeled with their corresponding v_i 's. In this example, $L = 16$, $n = 3$ and $(x_1, x_2, x_3) = (1, 6, 10)$.



Suppose now that D is formed by inserting all such (w, v_i) into it, i.e.

$$D = \{(w, v_i) \mid \overline{w} \subseteq (x_{i-1}, x_i] \wedge \overline{w'} \not\subseteq (x_{i-1}, x_i]\}.$$

Then D has the following useful properties (for proof see the Appendix):

Lemma 1.

1. If $(w, v_i) \in D$ and $(u, v_j) \in D$ then $w \not\prec u$.
2. For any $y \in [L]$ there is $(w, v_i) \in D$ such that $w \preceq y$.

We can now prove the following.

Corollary 1. Let $q \in (x_{i-1}, x_i]$. Then there is exactly one prefix $w \prec q$ for which $(w, v_i) \in D$. Every other prefix $p \prec q$ ($p \neq w$) is not contained as a keyword in D .

Proof. From part 2 of Lemma 1 it follows that there is some $w \prec q$ and some j such that $(w, v_j) \in D$. From part 1 of the same lemma it follows that w is indeed the only prefix of q which is contained in D . Finally, the construction of D is such that $\overline{w} \subseteq (x_{i-1}, x_i]$ so $i = j$. \square

User's query. On input $q \in [L]$ the user computes private keyword search queries kwq_1, \dots, kwq_l , where kwq_j corresponds to the prefix $p_j := q_1 || q_2 || \dots || q_j$. Corollary 1 ensures that exactly one of these queries will return the correct v_i .

Protocol Description The function corresponding to the interval labeling problem is

$$IL(q, (x_1, \dots, x_n, v_0, \dots, v_n)) = v_a, \text{ where } q \in (x_a, x_{a+1}].$$

Note that element rank

$$Rank(y, (x_1, \dots, x_n)) = |\{x_i \mid x_i < y\}|$$

is a special case of IL – use $v_i = i$ for $0 \leq i \leq n$. We show how to obviously compute IL , using private keyword search as the main building block. Let $(PKSG, PKSQ, PKSA, PKSD)$ be the algorithms implementing succinct computation of PKS . Then we have the following

Claim. Scheme 1 (see below) succinctly computes IL (i.e. it solves Problem 1).

Proof. Sketch. The main ingredient is the following simple claim.

Claim. Let $\overline{w} \subseteq (x_{i-1}, x_i]$ and $\overline{w'} \not\subseteq (x_{i-1}, x_i]$. Then $w \prec x_i \vee w \prec x_{i-1} \vee w' \prec x_i \vee w' \prec x_{i-1}$.

Proof. Straightforward case analysis. \square

Let D be the database obtained by the algorithm ILA . Using the above claim, it is easy to prove that $(w, v_i) \in D \iff (\exists i) \overline{w} \subseteq (x_{i-1}, x_i] \wedge \overline{w'} \not\subseteq (x_{i-1}, x_i]$. To see this, note that ILA iterates through the set $S = \{w \mid (\exists i) w \prec x_i \vee w' \prec x_i\}$; but the above claim guarantees that it is sufficient to examine exactly the set S . This means that Corollary 1 holds for D . We are now ready to prove:

Correctness. By Corollary 1, exactly one of the queries p_j has a matching entry $(p_j, v_i) \in D$, and i is such that $q \in (x_{i-1}, x_i]$.

Privacy. As said above, there is exactly one p_j with a matching database entry. All other $p_i \prec q$ will return \perp when used in private keyword search on D . Since the order of the answers is shuffled, using privacy of PKS and standard hybrid arguments, it is straightforward to show the privacy of interval labeling. For lack of space we do not present the proof here.

Efficiency. Note that D contains at most $2ln$ entries. Server computation is therefore $O(ln)$. Total communication, as well as user's computation is polylogarithmic in n (and linear in l).

Scheme 1 (Interval Labeling)

$ILG(1^k)$:

1. $output (SK, PK) = PKSG(1^k)$

$ILQ(SK, q)$:

1. $kw_{q_1} \leftarrow PKSQ(SK, q_1),$
 $\dots,$
 $kw_{q_l} \leftarrow PKSQ(SK, q_1 || \dots || q_l)$
2. $output (kw_{q_1}, \dots, kw_{q_l})$

$ILA(PK, (kw_{q_1}, \dots, kw_{q_l}), A = (v_0, x_1, v_1, \dots, x_n, v_n))$:

1. $for j = 1 to l:$
 - (a) $for i = 1 to n:$
 - i. $let w$ be the j -bit prefix of x_i
 - ii. $if \overline{w} \subseteq (x_{a-1}, x_a]$ and $\overline{w'} \not\subseteq (x_{a-1}, x_a]$ then add (w, v_a) to D
 - iii. $if \overline{w} \not\subseteq (x_{b-1}, x_b]$ and $\overline{w'} \subseteq (x_{b-1}, x_b]$ then add (w', v_b) to D
2. $if fewer than 2ln entries were added, then pad D to 2ln entries$
3. $for j = 1 to l, compute ra_j = PKSA(PK, kw_{q_j}, D)$
4. $randomly permute (ra_1, \dots, ra_l) and output the result$

$ILD(SK, (a_1, \dots, a_k))$:

1. $find the unique a_i such that PKSD(SK, a_i) = v \neq \perp; output v$

Note that the keywords which the algorithm ILA inserts in D need not be of the same length. PKS solutions do not support variable lengths, but it is easy to circumvent this problem: for example, a prefix-free encoding can be padded to some fixed length.

3.2 Extending to Higher Dimensions

Interval labeling can be naturally extended to labeling of (hyper-)rectangles in $[L]^d$. Here a point is defined by its d coordinates $(x_1, \dots, x_d) \in [L]^d$. Let $X_1 = (x_1[1] < \dots < x_1[n]), \dots, X_d = (x_d[1] < \dots < x_d[n])$ be d lists of coordinates (for simplicity assume they are all of equal length n). These lists then define $(n+1)^d$ rectangles, labeled $R_{0, \dots, 0}, \dots, R_{n, \dots, n}$. For $i_1, \dots, i_d \in [n]$ we define the rectangle $R_{i_1, \dots, i_d} := \{(x_1, \dots, x_d) \mid \forall j x_j \in (x_j[i_j], x_j[i_j+1]]\}$. In each rectangle R_{i_1, \dots, i_d} , a string label z_{i_1, \dots, i_d} is written.

This construction, which we will call *d-dimensional rectangle labeling*, is uniquely defined by

$(X_1, \dots, X_d, Z = (z_{0, \dots, 0}, \dots, z_{n, \dots, n}))$. We are interested in succinct computation of

$$RL_d(p = (x_1, \dots, x_d), (X_1, \dots, X_d, Z)) = z_{i_1, \dots, i_d} \text{ where } p \in R_{i_1, \dots, i_d}.$$

This problem can be solved by recursively using the succinct computation of interval labeling. Rather than write the general solution for any d , we will exemplify it using $d = 4$. This conveys the main idea clearly, but cumbersome notations associated with arbitrary d are avoided.

We will denote the $d = 4$ coordinates with (x, y, u, v) . Server's input is $X = (x_1, \dots, x_n)$, $Y = (y_1, \dots, y_n)$, $U = (u_1, \dots, u_n)$, $V = (v_1, \dots, v_n)$ ($x_i, y_i, u_i, v_i \in [L]$) and a table of values $Z = (z_{0,0,0,0}, \dots, z_{n,n,n,n})$ ($z_{i,j,k,l} \in \{0, 1\}^m$). User's input is a point $p = (x, y, u, v) \in [L]^4$. A scheme for succinct computation of RL_4 follows.

Scheme 2
 $RLG_4(1^k)$:

1. $output (PK, SK) = ILG(1^k)$

$RLQ_4(SK, q)$:

1. $output (ILQ(SK, x), ILQ(SK, y), ILQ(SK, u), ILQ(SK, v))$

$RLA_4(PK, (ex, ey, eu, ev), (X, Y, Z))$:

1. for $(i, j, k) \in [n]^3$ let $D_{i,j,k} \leftarrow ILA(PK, eqv, (v_1, z_{i,j,k,1}, \dots, v_d, z_{i,j,k,d}))$
2. for $(i, j) \in [n]^2$ let $E_{i,j} \leftarrow ILA(PK, equ, (u_1, D_{i,j,1}, \dots, u_c, D_{i,j,c}))$
3. for $i \in [n]$ let $F_i \leftarrow ILA(PK, eqy, (y_1, E_{i,1}, \dots, y_b, E_{i,b}))$
4. let $ans \leftarrow ILA(PK, eqx, (x_1, F_1, \dots, x_n, F_n))$
5. $output ans$

$RLD_4(SK, a)$:

1. $User outputs ILD(SK, ILD(SK, ILD(SK, ILD(SK, a))))$

Claim. Scheme 2 succinctly computes the function RL_4 .

4 Range Retrieval in 2 Dimensions

In this section we are dealing with range retrieval in 2-dimensions. For two points $a = (x_a, y_a)$, $b = (x_b, y_b)$ define the *rectangle* $\langle a, b \rangle$ as $\{(x, y) | x \in [x_a, x_b], y \in [y_a, y_b]\}$. The *server* holds a database D of n points p_1, \dots, p_n where $p_i \in [0, L] \times [0, L]$ (L is a public parameter). The *user* holds two points $q_1 = (qx_1, qy_1)$, $q_2 = (qx_2, qy_2) \in [0, L] \times [0, L]$, and he wishes to retrieve all the points $p_i \in D$ which are in the rectangle $\langle q_1, q_2 \rangle$. The communication of the protocol should be proportional to the number of retrieved points $|D \cap \langle q_1, q_2 \rangle|$, and the server should learn only that number.

The basic idea behind the protocol is as follows. Assume for simplicity that the server's database contains no two points with equal either x or y coordinates, i.e. $x_i \neq x_j$ and $y_i \neq y_j$ for all $i \neq j$. Define the *grid alignment* for a point $q = (x, y)$, denoted $Align(q)$ as the pair (i, j) where $i = |\{x_i | x_i < x\}|$ and $j = |\{y_i | y_i < y\}|$. Then, for any two rectangles $\langle a, b \rangle$, and $\langle a', b' \rangle$ we have that $D \cap \langle a, b \rangle = D \cap \langle a', b' \rangle$ if and only if $Align(a) = Align(a')$ and $Align(b) = Align(b')$. Therefore, the grid alignments of two corners of a rectangle, uniquely determine the points that lie inside that rectangle. Our protocol retrieves the points inside a rectangle by (1) determining the grid alignments corresponding to that rectangle by running the RL_4 protocol (2)

retrieving the points associated to those alignments by running a contiguous-PIR protocol. The assumption that no two database points agree in either coordinate can be easily removed (for example x -coordinate of i -th point could be shifted right by some amount $\delta_i < 1$, where all δ_i 's should be distinct; shifting by amount < 1 can be simulated by multiplying coordinates by a sufficiently large number; the same is applied to y -coordinates).

The protocol to retrieve the 2-d range is presented next. If the range size (the number of retrieved points) is s , and k denotes the security parameter then its communication complexity is proportional to $s \cdot \text{poly}(k)$, as is the user-side computation. Server-side computation is, however, proportional to $O(n^5)$.

2-d Range Retrieval Protocol

Range – $2(q_1, q_2; D)$:

User's input: points $(q_1 = (qx_1, qy_1), q_2 = (qx_2, qy_2))$

Server's input: database $D = (p_1 = (x_1, y_1), \dots, p_n = (x_n, y_n))$

User's Output: list of all p_i such that $x_i \in [qx_1, qx_2]$ and $y_i \in [qy_1, qy_2]$

- Server creates auxiliary databases X_1, Y_1, X_2, Y_2, Z and F as follows:
 - $X_1, X_2 \leftarrow (x_1, x_2, \dots, x_n), Y_1, Y_2 \leftarrow (y_1, y_2, \dots, y_n)$
 - set $t = 0$
 - for $(i, j, k, l) \in [n]^4$:
 - * compute s , the number of points in $\langle (x_i, y_j), (x_k, y_l) \rangle \cap D$
 - * $z_{i,j,k,l} \leftarrow (t, s)$
 - * store the s points in $\langle i, j \rangle \cap D$ in F at addresses $t \dots t + s - 1$
 - * $t \leftarrow t + s$
- User and server engage in a private computation of $RL_A((qx_1, qy_1, qx_2 + 1, qy_2 + 1); X_1, Y_1, X_2, Y_2, Z)$ (see section 3.2); user retrieves (t, s) in the end
- User and server engage in ContPIR (see section 2) with user's input (t, s) and server's input F

5 Private m^{th} -Ranked Element

Given two parties A and B with databases D_A and D_B , consider the problem of secure computation of the m^{th} -ranked element of $D_A \cup D_B$. In [1], Aggrawal et al. design a protocol for this problem with $O(\log(m))$ rounds and sublinear communication complexity in the size of database. We show how to use our non-interactive private element rank protocol along with additional tricks to design a protocol with better round efficiency. Particularly, we design a protocol that requires only $O(\log \log(m))$ rounds and has a communication complexity lower than $O(m^\epsilon)$ for any $0 \leq \epsilon \leq 1$.

In what follows we assume that the output of private element rank protocol is shared between the parties (no single party learns the actual output). This can be achieved by slightly modifying the protocol of section 3.1. More specifically, in the protocol of section 3.1, the server initially XORs all the interval labels by a random string r , and then the protocol proceeds as before. Server's share of the output will be r , while the user will receive the intended label XORed with r .

The underlying algorithm for our protocol is different from the one used by Aggrawal et al. The general idea for the algorithm is that each party searches for the m^{th} -ranked element in his/her list. One party will return the m^{th} -ranked element as a result

of the search. Next we show how to perform such a search securely and in a round- and communication-efficient manner. More specifically, we design a protocol with $O(c)$ rounds of interaction and $O(cm^{1/c})$ communication complexity for any $1 < c < n$. If we let $c = \log \log(m)$, we achieve a protocol with $O(\log \log(m))$ rounds and communication complexity of $O(m^{1/\log \log(m)})$ which is smaller than $O(m^\epsilon)$ for any $0 \leq \epsilon \leq 1$. It is not clear to us how to apply similar techniques to the protocol of [1] in order to achieve the same round efficiency.

Consider the first m elements of each database as our new databases. Initially, each database is divided into $m^{1/c}$ chunks of size $O(m^{(1-1/c)})$. In each iteration, we determine which chunk the m^{th} -ranked element belongs to by running the element rank protocol on the database elements that separate the chunks. We then repeat the same process on the chunk that contains the m^{th} -ranked element. The number of iterations required in order to find the m^{th} -ranked element is c . Next we describe the protocol in more detail:

Private m^{th} -ranked Element

Party A's input: database D_A .

Party B's input: database D_B .

Output: the m^{th} -ranked element in $D_A \cup D_B$.

1. Let's assume that databases D_A and D_B are sorted in an increasing order. Denote the set of first m elements of D_A and D_B by S_A and S_B , respectively. If either database is of size less than m , the database holder pads his/her database with values of $+\infty$ in order to get a database of size m . Finally, for every $1 \leq i \leq m^{1/c}$, let

$$r_A^i = r_B^i = i \times m^{(1-1/c)}$$

2. For $j = 1$ to c
 - (a) For $i = 1$ to $m^{1/c}$
 - i. Parties run the private element rank protocol to compute shares of:

$$r_A^{j,i} = \text{Rank}(S_A[r_A^i], S_B)$$

$$r_B^{j,i} = \text{Rank}(S_B[r_B^i], S_A)$$
 - ii. Let $R_A^i = r_A^i + r_A^{j,i}$ and $R_B^i = r_B^i + r_B^{j,i}$.
 - (b) Parties securely compute a circuit that takes shares of $r_A^i, r_A^{j,i}, r_B^i,$ and $r_B^{j,i}$ for every $1 \leq i \leq m^{1/c}$ as inputs and outputs shares of O_j where

$$O_j = S_A[r_A^i] \text{ if } R_A^i = m$$

$$O_j = S_B[r_B^i] \text{ if } R_B^i = m$$

$$O_j = 0 \quad \text{otherwise}$$

The circuit also outputs shares of the new r_A^i and r_B^i for every $1 \leq i \leq m^{1/c}$, where $r_A^i = R_A^{t_A} + i \times m^{(1-(j+1)/c)}$ and $r_B^i = R_B^{t_B} + i \times m^{(1-(j+1)/c)}$ and t_A is the largest integer such that $R_A^{t_A} \leq m$ (similarly for t_B).

3. Parties securely compute a circuit that takes shares of O_1, \dots, O_c and outputs shares of $\sum O_i$.

The circuits in steps 2(b) and 3 of the protocol are implemented using Yao’s garbled circuit protocol. The circuit for step 2(b) is of size $O(m^{1/c})$, and the circuit of step 3 is of size $O(c)$. The communication complexity of the protocol is dominated by step 2(b) which is repeated c times and is therefore $O(cm^{1/c})$. The round complexity of the protocol is c due to the number of iterations of the for loop.

6 One-round PIR Is Sufficient for Private Keyword Search

We recall the setup for private keyword search (PKS) problem. The server and the client are involved. The server’s input is a database X of n pairs (x_i, p_i) , where x_i is a keyword, and p_i the corresponding payload. The client’s input is a searchword w . If there is a pair (x_i, p_i) in the database such that $x_i = w$, then the output is the corresponding payload p_i . Otherwise the output is a special symbol \perp .

In [11], Freedman et al. give a non-interactive and communication efficient solution for this functionality. Their construction requires PIR, as well as a *semantically secure homomorphic encryption scheme*. In this section we design a modified version of their scheme, which eliminates the dependency on homomorphic encryption. In fact, any one-round PIR suffices for our solution, and the communication complexity is essentially the same as that of the underlying PIR. This establishes a somewhat surprising implication:

Theorem 3. *If there is a one-round PIR with communication complexity $c(n)$ then there is a one-round PKS with communication complexity $c(n)\text{polylog}(n)$.*

The basic idea of our solution is to replace the polynomials used in protocol of [11], by Yao’s garbled circuits [17]. This already eliminates the need for homomorphic encryption which was used to encrypt the coefficients of those polynomials. Further care is necessary to make sure that after this modification the protocol remains non-interactive. We use techniques similar to those used by Cachin et al. [4] for this purpose. For more details, please see the appendix.

Finally, note that all the protocols from previous sections can be implemented from PKS only, and obviously PIR is a special case of PKS. Thus the following three problems are “equivalent up to $\text{polylog}(n)$ factor in communication” (i.e. if any of them can be succinctly computed, then the others can too): PIR, PKS, interval labeling.

Acknowledgment: We would like to thank Yuval Ishai for providing us with a copy of [15], pointing to some related work and other helpful discussions.

References

1. Gagan Aggarwal, Nina Mishra, and Benny Pinkas. Secure computation of the k^{th} -ranked element. In *Proc. of Eurocrypt*, 2004.
2. Bill Aiello, Yuval Ishai, and Omer Reingold. Priced oblivious transfer: How to sell digital goods. In *Proc. of Eurocrypt*, 2001.

3. Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. One-way functions are essential for single-server private information retrieval. In *Proc. of the 31st ACM Symp. on the Theory of Computing (STOC)*, 1999.
4. Christian Cachin, Jan Camenisch, Joe Kilian, and Joy Mueller. One-round secure computation and secure autonomous mobile agents. In *ICALP '00: Proceedings of the 27th International Colloquium on Automata, Languages and Programming*, pages 512–523, London, UK, 2000. Springer-Verlag.
5. Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *Proc. of Eurocrypt*, pages 402–414, 1999.
6. Yan-Cheng Chang. Single database private information retrieval with logarithmic communication. *Cryptology ePrint Archive, Report 2004/036*, 2004.
7. B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. *Technical Report TR CS0917, Department of Computer Science, Technion*, 1997.
8. Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *Proc. of FOCS*, pages 41–50, 1995.
9. Giovanni Di Crescenzo, Tal Malkin, and Rafail Ostrovsky. Single database private information retrieval implies oblivious transfer. In *Proc. of the 18th Annual Eurocrypt conference*, 2000.
10. Nenad Dedić, Leonid Reyzin, and Scott Russell. Unpublished manuscript.
11. Michael J. Freedman, Yuval Ishai, Benny Pinkas, and Omer Reingold. Keyword search and oblivious pseudorandom functions. In *Second Theory of Cryptography Conference (TCC 2005)*, pages 303–324, 2005.
12. Yael Gertner, Yuval Ishai, Eyal Kushilevitz, and Tal Malkin. Protecting data privacy in private information retrieval schemes. In *Proc. ACM STOC*, 1998.
13. O. Goldreich. Foundations of cryptography - volume 2. 2004.
14. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *proceedings of 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.
15. Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. *To appear at Theory of Cryptography Conference (TCC)*, 2007.
16. Eyal Kushilevitz and Rafail Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of FOCS*, pages 364–373, 1997.
17. Yehuda Lindell and Benny Pinkas. A proof of Yao’s protocol for secure two-party computation. *Cryptology ePrint Archive, Report 2004/175*, 2004.
18. Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In *Information Security Conference (ISC05)*, volume volume 3650 of Lecture Notes in Computer Science, pages 314–328, 2005.
19. Moni Naor and Kobbi Nissim. Communication preserving protocols for secure function evaluation. In *pot 33rd STOC*, pages 590–599, 2001.
20. Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. ACM STOC*, 1999.
21. Rafail Ostrovsky and William E. Skeith. Private searching on streaming data. In *Proc. of CRYPTO*, pages 223–240, 2005.
22. Michael O. Rabin. How to exchange secrets by oblivious transfer. technical report tech. In *Technical Report Tech. Memo TR-81*, 1981.
23. Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC 1. In *IEEE Symposium on Foundations of Computer Science*, pages 554–567, 1999.

A Non-interactive Private Keyword Search Based on PIR

Here we follow the notation introduced in section 2.1 for succinct computation. Let $(PIRG, PIRQ, PIRA, PIRD)$ be the algorithms implementing the succinct computation of PIR and (OTG, OTQ, OTA, OTD) be the algorithms implementing succinct computation of 1-out-of-2 oblivious transfer. Our private keyword search protocol follows:

Scheme 4 (Private Keyword Search)

$PKSG(1^k)$:

1. Compute $(SK_1, PK_1) \leftarrow PIRG(1^k)$ and $(SK_2, PK_2) \leftarrow OTG(1^k)$
2. Output $(SK = \{SK_1, SK_2\}, PK = \{PK_1, PK_2\})$

$PKSQ(SK, w)$:

1. $otq_1 \leftarrow OTQ(SK_2, w_1)$,
 $otq_2 \leftarrow OTQ(SK_2, w_2)$,
 \dots ,
 $otq_k \leftarrow OTQ(SK_2, w_k)$
2. $pirq \leftarrow PIRQ(SK_1, H(w))$, where H is public hash function
3. output $(pirq, otq_1, \dots, otq_k)$

$PKSA(PK, (pirq, otq_1, \dots, otq_k), X = (x_1, p_1, \dots, x_n, p_n))$:

1. The server defines L bins and maps the n items into the L bins using a random, publicly-known hash function H with a range of size L . H is applied to the database's keywords, i.e., (x_i, p_i) is mapped to bin $H(x_i)$. Let m be a bound such that, with high probability, at most m items are mapped to any single bin. (L and m are parameters to be determined later.)
2. For every bin j , the server constructs a Yao's garbled circuit GC_j . Circuit GC_j takes as input: (1) Server's: every pair (x_i, p_i) mapped to bin j , (2) Client's: keyword w , and outputs p_i if $w = x_i$, for any x_i fed to the circuit and \perp symbol otherwise. We require that server uses the same garbled values for client's input wires in all circuits and denote them by $(g_1^0, g_1^1), \dots, (g_k^0, g_k^1)$.^a Let $B = \{GC_1, \dots, GC_L\}$.
3. Compute $pira = PIRA(PK_1, pirq, B)$ and $ota_i = OTA(PK_2, otq_i, (g_i^0, g_i^1))$ for $1 \leq i \leq k$
4. Output $(pira, ota_1, \dots, ota_k)$

$PKSD(SK, (pira, ota_1, \dots, ota_k))$:

1. Client computes $PIRD(SK_1, pira)$ to get the garbled circuit $GC_{H(w)}$, and computes $OTD(SK_2, ota_i)$ for $1 \leq i \leq k$ to get his garbled inputs for the circuit. He then evaluates the circuit and computes the output using the translation table

^a The garbled circuit also includes garbled values for server's input wires and a translation table that enables the client to learn the final output. Please see [17] for more detail on Yao's garbled circuit construction.

Letting $L = n/\log(n)$ will lead to garbled circuits of size $O(\text{polylog}(n))$, and hence, the total communication complexity of $O(\text{polylog}(n))$ for the above protocol. The protocol requires one round of communication between the server and the client.

The security follows from security of Yao's garbled circuit protocol (pseudorandom functions and OT), and the security of the PIR scheme used. From results of [9, 3] we know that one-round PIR implies pseudorandom functions and one-round 1-out-of-2 OT. Hence based on the above construction, *non-interactive* PIR implies *non-interactive* PKS. Furthermore, it is easy to implement non-interactive PIR given non-interactive PKS by simply letting the keywords be the indices of the values in the database. Therefore we can conclude that *non-interactive PIR and non-interactive PKS are equivalent*. This automatically proves equivalence of non-interactive interval labeling and rectangle labeling problems to non-interactive PIR as well.

B Proof of Lemma 1

1. Suppose $w \prec u$. Since $(u, v_j) \in D$ we have that $\bar{u} \subseteq (v_{j-1}, v_j]$ and $\bar{u}' \not\subseteq (v_{j-1}, v_j]$. On the other hand since $(w, v_i) \in D$, we have $\bar{w} \subseteq (v_{i-1}, v_i]$. But from $w \prec u$ it follows that $\bar{u} \cup \bar{u}' \subseteq \bar{w}$. \bar{w} intersects both $(v_{i-1}, v_i]$ and $(v_{j-1}, v_j]$, so it must be that $i = j$. But then $\bar{u}' \subseteq \bar{w} \subseteq (v_{j-1}, v_j]$ – contradiction!
2. Consider $S = \{w \mid w \preceq y \wedge (\forall i)w \not\prec x_i\}$. Let w be the shortest string in S . Then for some i , $(w, v_i) \in D$. Indeed, since w is the shortest string in S , either $w = \epsilon$ or $w^- \notin S$. If $w = \epsilon$ then $n = 0$, so $(\epsilon, v_0) \in D$. If $w^- \notin S$, then $w^- \prec x_i$ for some i , so it must be that $w' \prec x_i$. This means that $\bar{w} \subseteq (x_{i-1}, x_i]$ but $\bar{w}' \not\subseteq (x_{i-1}, x_i]$. So it must be that $(w, v_i) \in D$.