Privacy-Preserving Range Queries from Keyword Queries

Giovanni Di Crescenzo, Abhrajit Ghosh

Applied Communication Sciences, NJ, USA.
{gdicrescenzo,aghosh}@appcomsci.com

Abstract. We consider the problem of a client performing privacy-preserving *range queries* to a server's database. We propose a cryptographic model for the study of such protocols, by expanding previous well-studied models of keyword search and private information retrieval to the range query type and to incorporate a multiple-occurrence attribute column in the database table.

Our first two results are 2-party privacy-preserving range query protocols, where either (a) the value domain is linear in the number of database records and the database size is only increased by a small constant factor; or (b) the value domain is exponential (thus, essentially of arbitrarily large size) in the number of database records and the database size is increased by a factor logarithmic in the value domain size. Like all previous work in private information retrieval and keyword search, this protocol still satisfies server time complexity linear in the number of database payloads.

We discuss how to adapt these results to a 3-party model where encrypted data is outsourced to a third party (i.e., a cloud server). The result is a private database retrieval protocol satisfying a highly desirable tradeoff of privacy and efficiency properties; most notably: (1) *no unintended information* is leaked to clients or servers, and the information leaked to the third party is characterized as 'access pattern' on encrypted data; (2) for each query, all parties run in time *only logarithmic* in the number of database records and linear in the answer size; (3) the protocol's query runtime is practical for real-life applications.

1 Introduction

The recent computing trend of outsourcing big data in the cloud for simplified and efficient application deployment is being embraced in government, as well as other areas, including finance, information technology, etc. In government, large databases are needed in many contexts (e.g., no-fly lists, metadata of communication records, etc.). In finance, banks and other financial institutions need to store huge data volumes and compute over them on a daily basis. In information technology, web and social networks collect huge data from computer users, which is then made available for different uses and computations. To facilitate and guarantee success for all of these applications, databases are very useful data management tools, and cloud storage and computing provide

tremendous efficiency and utility for users, as exemplified by the increasingly successful database-as-a-service application paradigm (see, e.g., [13]). On the other hand, cloud storage and computing paradigms are also accompanied by privacy risks (see, e.g., [21]). To mitigate these risks, database-management systems can use *privacy-preserving database retrieval protocols* that allow users to submit queries and receive results in a way that clients learn nothing about the contents of a database except the results of their queries, and servers do not learn which queries are submitted. The research literature has attempted to address these issues, by studying private database retrieval protocols in limited database and query models and with limited efficiency properties. In this paper we partially address some of these limitations, by using a practical database model, and proposing protocols in both a client-server model and a 3-party model, where servers can outsource data to a third party (in encrypted form). In these models, practical and privacy-preserving database retrieval protocols for basic query types such as keyword queries, have been recently shown to be possible. In this paper, we attempt to show that practical and privacy-preserving database retrieval protocols are possible for a more complex query type: *range queries*.

Previous work. The security and cryptography literature contains a significant amount of research in the private information retrieval (PIR) [6, 18, 20] and keyword search (KS) [5, 3, 10] areas. Both areas consider rather theoretical data models, as we now discuss. In PIR, a database is modeled as a string of n bits, and the query value is an index $i \in \{1, \ldots, n\}$. In KS, early data models were also somewhat restrictive; for instance, [10] only admitted a single matching record per query. The inefficiency of the server runtime in PIR and KS protocols has been well documented (see, e.g., [24]). Some results attempted to use a third party and make the PIR query subprotocol more efficient but require a practically inefficient preprocessing phase [8]. Recently, however, some results on provably privacy-preserving and practical keyword queries in a practical database model and in an outsourced-data scenario were concurrently shown by [7, 17], where significant efficiency is achieved by provably limiting the privacy loss to encrypted data "access-pattern" information, only leaked to the cloud server.

The literature also contains a significant amount of work on range queries or range computations on encrypted data. Some papers (starting with [22, 4]) focus on encrypting messages, on which one can later perform range query computations. These approaches offer interesting provable security properties but make heavy use of asymmetric cryptography techniques and seem hard to translate into practical protocols for databases. Promising approaches to achieve at least some limited amount of privacy (with tradeoffs against efficiency) on range queries in an outsourced database setting have also been shown (see, e.g., [14] and follow-up work), typically based on variants of "bucketization" approaches. The primitive of order-preserving encryption gives rise to elegant and efficient range query protocols in the "database-as-a-service" model (see, e.g., [1] and follow-up work), but constructions of order-preserving encryption are still not very efficient and especially come with static leakage on the encrypted data to

the server holding it [2]. Overall, the question of designing provably privacy-preserving range queries in a practical data model, even in the outsourced-data scenario, seems to still deserve more attention from the security community.

Our contribution. We study range queries in a more practical (outsourced or not) database model, capturing record payloads, possibly equal attribute values across different database records, and multiple answers to a given query. In this model, we define suitable correctness, privacy and efficiency requirements.

We then design two range query protocols in the 2-party model, which satisfy desired privacy properties (i.e., the server learns no information about the query range other than the number of matching records, and the client learns no information about the database other than matching database records) in our data model. Our first protocol works for linear-size value domains by only increasing database size by a small constant, and our second protocol works for exponential-size (thus, essentially arbitrary-size) value domains while increasing database size by a factor logarithmic in the value domain size. These protocols are constructed directly from any KS protocol and, like previous PIR and KS protocols, have server time complexity linear in the database size, a drawback dealt with in our next result.

Our third protocol transforms any of our 2-party range query protocols into a 3-party protocol, where the third party can be a cloud server, based on any 3-party KS protocol (like the one in [7], only based on any pseudo-random function, implemented as a block cipher). In this protocol, both server and third party *run queries in logarithmic time* and the following privacy properties provably hold: the server learns nothing about the query range, the client learns nothing about the database in addition to the matching database records, and the third party learns nothing about the query range or the database content, other than the repeating of queries from the client and repeated access to the encrypted data structures received by the server at initialization. This solves the problem of achieving provable privacy (against a semi-honest adversary) and efficient server runtime at the cost of a 'third-party'-server and some leakage to the third party characterized as 'access-pattern' to encrypted data. We stress that this protocol has efficient running time not only in an asymptotic sense, but in a sense that makes it ready for real-life applications (where such form of leakage to the third party is tolerable). In our implementation of a computationally similar protocol, we reached our main performance goal of achieving response time to be *less than 1 order of magnitude* slower than commercial non-private protocols like MySQL. Our protocol solves a number of technical challenges using simple and practical techniques, including a reduction step via an intermediate rank database and a 'lazy' database value shifting approach. The privacy loss traded for such a practicality property was already studied in [15, 16], who also proposed simple techniques to mitigate leakage to the cloud server in the form of 'access-pattern' to encrypted data, at least in the case of keyword queries. (Here, note that in the presence of such leakage, neither the client nor the server learn anything new, and the cloud server does not statically learn anything about the plain database content.) We believe that one appropriate mitigation technique needed

for such solutions could be based on Oblivious RAM (an active area started in [12]), and it is plausible that dedicated Oblivious RAM techniques in the 3-party model may nullify or mitigate any such leakage based on 'access-pattern' over encrypted data. This is indeed a promising direction as, while years ago Oblivious RAM was considered inefficient, recent advances (see, e.g., [23]) have made it significantly less inefficient. In all our protocols, we only consider privacy against a semi-honest adversary corrupting at most one party (i.e., an adversary that follows the protocol and then attempts to violate the privacy of one of the parties).

2 Models and Requirements

Data and Query Models. We model a *database* as an n-row, 2-column matrix $D = (A_1, A_2)$, where each column is associated with an *attribute*, denoted as A_j, for $j = 1, 2$, and each *entry* is denoted as $A_j(i)$. The first column is a *value attribute*, where entries are values in a *domain Dom* with a total order $\leq$, and the last column A_2, is a *payload attribute*, where entries can be arbitrary binary strings. The database *schema*, assumed to be publicly known to all parties, includes parameter n, the security parameter, and the description of the attribute value domain. A database row is also called *record*, and is assumed to have the same length ℓ_r (if data is not already in this form, techniques from [9] are used to efficiently achieve this property), where ℓ_r is constant with respect to n.

A *query* q is modeled to contain one or more *query values* from the relative attribute domains. We mainly consider Range queries, defined as:

$$\text{SELECT } * \text{ FROM } main \text{ WHERE attribute\_name} \in [v_0, v_1],$$

where v_0, v_1 are the query values. A *valid response (to a range query)* consists of all payloads $A_2(i)$, for $i \in [1, n]$, such that $A_1(i) \in [v_0, v_1]$, and we say that these payloads (or records) *match* the query. We also discuss KS queries, defined as:

$$\text{SELECT } * \text{ FROM } main \text{ WHERE attribute\_name} = v,$$

where v is the query value. A *valid response (to a keyword query)* consists of all payloads $A_2(i)$, for $i \in [1, n]$, such that $A_1(i) = v$.

Participant Models. We consider the following *efficient* (i.e., running in probabilistic polynomial-time in a common security parameter 1^σ) participants. The *client* is the party, denoted as C, that is interested in retrieving data from the database. The *server* is the party, denoted as S, holding the database (in the clear), and is interested in allowing clients to retrieve data. The *third party*, denoted as TP, helps the client to carry out the database retrieval functionality and the server to satisfy efficiency requirements during the associated protocol. By *2-party model* we denote the participant model that includes C, S and no third party. By *3-party model* we denote the participant model that includes C, S, and TP. (See Figure 1,2 for a comparison of the two participant models.)

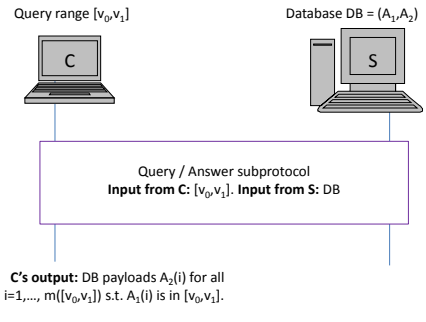

Fig. 1. Structure of our 2-party RQ protocol

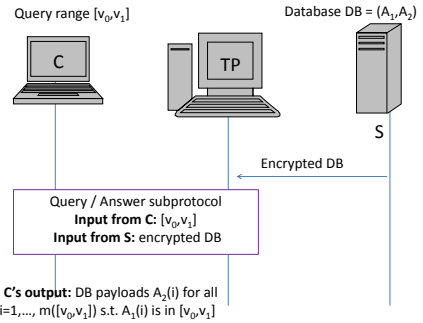

Fig. 2. Structure of our 3-party RQ protocol

Range query protocols. In the above data, query, and participant models, we consider a *(static-data) range query* (briefly, RQ) protocol that extends the KS protocol, as defined in [10] (in turn, an evolution of the PIR protocol, as defined in [18]), in that it considers range queries instead of keyword queries, and it allows the attribute column to have multiple occurrences of the same value. (We can also extend the model so to incorporate databases that contain multiple attributes). Specifically, we define an RQ protocol as a pair $(\mathsf{Init}, \mathsf{Query})$ of subprotocols, as follows. The initialization subprotocol Init is used to set up data structures and cryptographic keys before C's queries are executed. The query subprotocol Query allows C to make a single query to retrieve (possibly multiple) matching database records. We also define an *RQ protocol execution* as a sequence of executions of subprotocols $(\mathsf{Init}, \mathsf{Query}_1, \ldots, \mathsf{Query}_q)$, for some q polynomial in the security parameter, and all subprotocols are run on inputs provided by the involved parties (i.e., a database from S and query values from C). We would like to build RQ protocols that satisfy the following (informal) list of *requirements*:

1. *Correctness*: the RQ protocol allows a client to obtain all payloads from the current database associated with records that match its issued query; more specifically, for any RQ protocol execution, and any inputs provided by the participants, in any execution of a Query subprotocol, the probability that C obtains all records in the current database that match C's query value input to this subprotocol, is 1.
2. *Privacy*: informally speaking, the RQ protocol preserves privacy of database content and query values, ideally only revealing what is leaked by system parameters known to all parties and by the intended functionality output (i.e., all payloads in matching records to C); more specifically, we require the subprotocols in an RQ protocol execution to not leak information beyond the following
   - Init: all system parameters, including the database schema and a security parameter, will be known to all participants; in the 3-party model, an

additional string *eds* (for encrypted data structures) will be known to TP, will be encrypted under one or more keys unknown to TP and its length is known from quantities in the database schema;

- Query, based on query range $qr = [v_0, v_1]$ and the database D: all payloads $\{p(i) : i = i(1), \ldots, i(m(qr))\}$ such that $A_1(i) \in [v_0, v_1]$, for $i = i(1), \ldots, i(m(qr))$, will be obtained by C, as a consequence of the correctness requirement; in the 2-party model, the value $m(qr)$ will be known to S; in the 3-party model, the value $m(qr)$, all bits in *eds* read by TP according to the instructions in the Query protocol, and which previous executions of Query used the same query value v, will be known to TP.

3. *Efficiency*: the protocol should have low time, communication and round complexity, as a function of system parameters, including the number n of database records.

Given the characterization of intended leakage in the above privacy definition, a formal privacy definition can be derived using known definition techniques from simulation-based security and composable security frameworks often used in the cryptography literature.

Similarly as noted for keyword queries in [7], we observe that the communication exchanged in each execution of any subprotocol Query has to leak an upper bound on the value $m(qr)$, i.e., the number of matching records, to S in the 2-party model, and to the coalition of TP and S in the 3-party model. Accordingly, we target the design of protocols that may leak $m(qr)$ to S in the 2-party model. In the 3-party model, different RQ protocols could leak $m(qr)$ only to S, or only to TP, or somehow split this leakage between S and TP. Having to choose between one of these options, we made the practical consideration that privacy against S (i.e., the data owner) is typically of greater interest than privacy against TP (i.e., the cloud server helping C retrieve data from S) in many applications, and therefore we focused in this paper on seeking protocols that leak $m(qr)$ to TP and nothing at all to S. Moreover, in the 3-party model, we made a definitional choice of leaking patterns of repeated access to encrypted data to TP; this is not due to a theoretical limitation, but seems a well-characterized privacy leakage, which, depending on the application at hand, either is a small price to pay towards achieving very efficient time-complexity requirements on S and TP, or can be reduced by using separate techniques.

With respect to efficiency, although we design protocols with low time, round and communication complexity, we focus our discussions on the communication complexity of the query subprotocols, and on the running time of S in the 2-party model and of S and TP in the 3-party model.

Background: Keyword Search protocols. A *random function* R is a function that is chosen with distribution uniform across all possible functions with some pre-defined input and output domains. A keyed function $F(k, \cdot)$ is a *pseudo-random function* (PRF, first defined in [11]) if, after key k is randomly chosen, no efficient algorithm allowed to query an oracle function O can distinguish whether O is $F(k, \cdot)$ or O is a random function R (over the same input and

output domain), with probability greater than 1/2 plus a negligible quantity. A *KS protocol* is a protocol between two parties A, having as input a keyword $v \in \{1, \ldots, n\}$, and B, having as input a 2-column database represented as $D = (A_1, A_2)$. The protocol consists in a private retrieval of the value(s) $A_2(i)$ such that $A_1(i) = v$, returned to A (thus, without revealing any information about i to B or about $A_2(1), \ldots, A_2(i-1), A_2(i+1), \ldots, A_2(n)$ to A). Several KS protocols have been presented in the cryptographic literature, starting with [18], using number-theoretic hardness assumptions (see also [5, 10, 7]).

3 Range Queries in the Two-Party Model

We describe two RQ protocols for range queries in this model: the first protocol, presented in Section 3.1, works for ranges with elements in any linear-size domain; the second protocol, presented in Section 3.2, works for ranges with elements in any exponential-size (in practice, arbitrarily large) domain.

3.1 A Range Query Protocol for Linear-Size Domains

Our first 2-party RQ protocol considers range values in linear-size domains (that is, where the domain size is equal to the number of database records). This protocol follows the general structure outlined in Figure 1 and satisfies the following

Theorem 1. Consider a database with n records and domain $Dom = [0, n-1]$. Assuming the existence of a 2-party privacy-preserving KS protocol $\pi_0 = (\mathsf{Init}_0, \mathsf{Query}_0)$, there exists (constructively) a 2-party privacy-preserving RQ protocol $\pi_1 = (\mathsf{Init}_1, \mathsf{Query}_1)$ for such a database, satisfying:

1. correctness
2. privacy against C (i.e., it only leaks the matching records to C);
3. privacy against S (i.e., it only leaks the number of matching records to S);
4. communication complexity of Query_1 on a queried range qr is $O(m(qr))$ times the communication complexity of Query_0;
5. the S-time complexity in Query_1 on a queried range qr is $O(m(qr))$ times the S-time complexity in Query_0 plus $O(n)$.

We prove Theorem 1 by describing RQ protocol π_1 and its properties.

The RQ protocol π_1: basic definitions. Let Dom be a value domain with a total order $\leq$ defined on it. We say that Dom is a *linear-size domain* if it holds that $|Dom| \leq n$. Given a list U of (not necessarily distinct) values $u_1, \ldots, u_n \in Dom$, we say that a value $v \in Dom$ has *lower U-rank* r, also denoted as $Lrank(U, v) = r$, if there are r values strictly smaller than v. We say that a value $v \in Dom$ has *upper U-rank* r, also denoted as $Urank(U, v) = r$, if there are $n - r$ values in U strictly larger than v. Let $sU = (u_{h(0)}, \ldots, u_{h(n-1)})$ denote the list obtained from U by sorting its n elements. These definitions directly imply the following:

Fact 1 Given values $v_0, v_1 \in Dom$ such that $v_0 \leq v_1$, it holds that

1. $U \cap [v_0, v_1] = \emptyset$ if and only if $Lrank(U, v_0) \geq Urank(U, v_1)$.
2. $U \cap [v_0, v_1] \neq \emptyset$ if and only if $u_{h(a)}, \ldots, u_{h(b)} \in [v_0, v_1]$, for $a = Lrank(U, v_0)$ and $b = Urank(U, v_1) - 1$.

The RQ protocol π_1: an informal description. A first approach in our protocol goes as follows. At initialization S splits database D into two databases: a rank database rD and a payload database pD. At query time, C asks S for the lower rank of v_0 and the upper rank of v_1, where $[v_0, v_1]$ denotes the range queried by C. Because in this protocol we consider only linear-size value domains, S can store at initialization the lower rank and the upper rank of each value in the domain in rD; thus, it suffices C to perform a keyword query to rD to retrieve the two upper and lower rank values. Given these retrieved values, C can compute how many attribute values (if any) are in $[v_0, v_1]$ (i.e., the upper rank minus the lower rank), and then perform as many keyword queries in pD to retrieve the records matching the queried range. As written so far, the protocol satisfies our desired correctness and efficiency properties, but not the privacy property, as C learns the two rank values associated with the queried range's endpoints. We fix this problem by requiring S to randomize the rank values by a random shift of the attribute values, a variation of an idea first used in [8] to improve the efficiency of keyword queries in a 3-party model. Thus, the ranks received by C will be randomly distributed, conditioned by the fact that the difference between them remains the same, and C is entitled to know this difference because of the correctness requirement.

The RQ protocol π_1: a formal description. Protocol π_1 uses a KS protocol $\pi_0 = (\mathsf{Init}_0, \mathsf{Query}_0)$ for a 2-column database, which can be obtained from protocol π_1 in [7] or protocol 2 in [10]. Both these protocols use the KS protocol in [5], which in turn is based on any semi-private PIR protocol (e.g., [18]).

Init_1. On input database $D = (A_1, A_2)$, S sets U as the list $(A_1(1), \ldots, A_1(n))$, and builds an associated rank database $rD = (rA_1, rA_2)$ and an associated payload database $pD = (pA_1, pA_2)$, computed as follows.
For each $i = 1, \ldots, n$,
1. $rA_1(i) = i$,
2. $rA_2(i) = (Lrank(U, i)), Urank(U, i))$, and

for each $i = 0, \ldots, n$,
1. $pA_1(i) = i$, and
2. $pA_2(i) = A_2(j)$ where $j \in \{1, \ldots, n\}$ satisfies $Lrank(U, A_1(j)) = i$.

Query_1. Let $\gg_n$ denote the operation 'right shift modulo n'. On input query range $qr = [v_0, v_1]$, where $v_0, v_1 \in Dom$, from C, and all quantities computed during Init_1, the following steps are run:
1. If $v_0 > v_1$ then C sends failure symbol $\perp$ to S and halts;
2. S randomly chooses value $s \in \{0, \ldots, n-1\}$
3. for $i = 1, \ldots, n$,
   S sets $Lr'(U, i) = Lrank(U, i) \gg_n s$, $Ur'(U, i) = Urank(U, i) \gg_n s$;
   S sets $rA'_2(i) = (Lr'(U, i), Ur'(U, i))$
4. S runs Init_0 on input 2-column database $rD = (rA_1, rA'_2)$

5. for $j = 0, 1$: C and S run $\mathsf{Query_0}$, where C uses v_j as query value and S provides (rA_1, rA'_2) as a 2-column database; at the end of the protocol, C computes the payload $rA'_2(i(j)) = (Lr'(U, i(j)), Ur'(U, i(j)))$ such that $rA_1(i(j)) = v_j$;
6. if $Lr'(U, i(0)) = Ur'(U, i(1))$ then
   C sends failure symbol $\perp$ to S and halts.
7. for $i = 0, \ldots, n$,
   S sets $pA'_1(i) = pA_1(i) \gg_n s$;
8. S runs $\mathsf{Init_0}$ on input 2-column database $pD = (pA'_1, pA_2)$
9. for $j = Lr'(U, i(0)), \ldots, Ur'(U, i(1)) - 1$, possibly cycling from $n-1$ to 0: C and S run $\mathsf{Query_0}$, where C uses j as query value and S provides (pA'_1, pA_2) as a 2-column database; at the end of the protocol, C computes the payload $pA_2(i(j))$ such that $pA'_1(i(j)) = j$.

Properties of π_1. We now show that π_1 satisfies the correctness, privacy and efficiency properties defined in the 2-party model.

Correctness. First of all, note that by the test in step 1, we can assume that $v_0 \leq v_1$, which implies that $Lrank(U, i(0)) \leq Urank(U, i(1))$.

By the correctness property of the KS protocol π_0, at the end of step 5 of $\mathsf{Query_1}$, C can compute the shifted lower rank $Lr'(U, i(0))$ of v_0 and the shifted upper rank $Ur'(U, i(1))$ of v_1. As both values are obtained as a shift, by the same random number s, of $Lrank(U, i(0))$ and $Urank(U, i(1))$, respectively, it holds that $Lr'(U, i(0)) = Ur'(U, i(1))$ if and only if $Lrank(U, i(0)) = Urank(U, i(1))$. Using item 1 of Fact 1, this implies that if $U \cap [v_0, v_1] = \emptyset$, it will hold that $Lrank(U, i(0)) = Urank(U, i(1))$ and thus $Lr'(U, i(0)) = Ur'(U, i(1))$, and then C will halt in step 6 of $\mathsf{Query_1}$, without receiving any payload from S. On the other hand, if $U \cap [v_0, v_1] \neq \emptyset$, at the end of step 9 of $\mathsf{Query_1}$, by the correctness property of the KS protocol π_0, C computes the payload $pA_2(i(j))$ such that $pA'_1(i(j)) = j$, for all $j = Lr'(U, i(0)), \ldots, Ur'(U, i(1)) - 1$, possibly cycling from $n-1$ to 0. Using item 2 of Fact 1, this implies that S receives all payloads corresponding to values $A_1(i)$ in the range $[v_0, v_1]$.

Privacy. We show that π_1 satisfies our privacy requirement when the adversary corrupts any one among S or C.

When the adversary corrupts S, privacy (i.e., corrupting S does not provide the adversary any new information about C's range query $[v_0, v_1]$ other than system parameters and the number of matching payloads) can be proved by using the analogue privacy property of the KS protocol π_0. First of all, we observe that $\mathsf{Query_1}$ in protocol π_1 consists of 1 execution of $\mathsf{Query_0}$ followed by either no further execution of $\mathsf{Query_0}$ (resulting in no payload received by C) or by $m(qr) = Urank(U, v_1) - Lrank(U, v_0)$ additional executions of $\mathsf{Query_0}$ (resulting in $m(qr) > 0$ payloads received by C). Thus, given the number $m(qr) \geq 0$ of payloads received by C, an efficient simulator for the view obtained by S is obtained by suitably calling the efficient simulator for the view by S in the KS protocol π_0.

When the adversary corrupts C, privacy (i.e., corrupting C does not provide the adversary with any information about S's database D other than system

parameters and what intended by the correctness requirement) can be proved by using the analogue privacy property of protocol π_0. Here, the proof is similar to the previous case: given the number $m(qr) \geq 0$ of payloads received by C, a simulator for C's view is obtained by suitably calling the simulator for C's view in the KS protocol π_0.

Efficiency. As Query_1 essentially consists of running $m(qr) + 1$ times Query_0, the communication complexity (resp., S-time complexity) of Query_1 is $O(m(qr))$ times the communication complexity (resp., S-time complexity) of Query_0. Thus, the communication complexity is desirably linear in the number of matching records (and can be sub-linear in the number n of total database records). Analogously, the S-time complexity of Query_1 is $O(m(qr))$ times the S-time complexity of Query_0 plus $O(n)$. Here, note that the S-time complexity of Query_1 is linear in n already for small values of $m(qr)$ as so is the S-time complexity of Query_0. This inefficiency is a major and known drawback of all 2-party model solutions for protocols like PIR, KS, and therefore, of protocols π_0 and π_1. Indeed, this motivated our study of RQ protocols in the 3-party model in Section 4.

3.2 A Range Query Protocol for Exponential-Size Domains

Our second 2-party RQ protocol considers range values in exponential-size (which means, practically speaking, arbitrarily large) domains. This protocol follows the general structure outlined in Figure 1 and satisfies the following

Theorem 2. Consider a database with n records and domain $Dom = [0, 2^d - 1]$, for some d polynomial in n. Assuming the existence of a 2-party privacy-preserving KS protocol $\pi_0 = (\mathsf{Init}_0, \mathsf{Query}_0)$, there exists (constructively) a 2-party privacy-preserving RQ protocol $\pi_2 = (\mathsf{Init}_2, \mathsf{Query}_2)$ for such a database, satisfying:

1. correctness
2. privacy against C (i.e., it only leaks the matching records to C);
3. privacy against S (i.e., it only leaks the number of matching records to S);
4. communication complexity of Query_1 on a queried range qr is $O(m(qr))$ times the communication complexity of Query_0;
5. the S-time complexity in Query_1 on a queried range qr is $O(m(qr))$ times the S-time complexity in Query_0 plus $O(dn)$.

We prove Theorem 2 by describing RQ protocol π_2 and its properties.

The RQ protocol π_2: basic definitions. Let Dom be a value domain with a total order $\leq$ defined on it. We say that Dom is an *exponential-size domain* if it holds that $|Dom| \leq 2^d \leq 2^{p(n)}$, for some polynomial p. For simplicity, we restrict to the case Dom is the *d-dimensional hypercube*, i.e. $Dom = [0, 2^d - 1]$, but note that our results can be extended to any exponential-size domain.

We define the *set $cI(Dom)$ of canonical intervals* for Dom by the following recursion: first, add Dom into $cI(Dom)$; then, split Dom into Dom_0, containing the first half of its elements, and Dom_1 containing the second half; then, for

$i = 0, 1$, generate $cI(Dom_0)$, the set of canonical intervals for Dom_i; finally, add $cI(Dom_0), cI(Dom_1)$ to $cI(Dom)$.

An interval $[a, b] \subseteq Dom$ is a *border interval in Dom* if there exists an interval $I \in cI(Dom)$ such that either a is the first element in I or b is the last element in I. The following fact directly follows by the above definitions of border and canonical intervals.

Fact 2 For every interval $[a, b] \subseteq Dom$, either $[a, b]$ is a border interval in Dom, or there exists c such that $[a, c]$ and $[c + 1, b]$ are border intervals in Dom.

For any interval $[a, b] \subseteq Dom$, intervals $[a, c_1], [c_1 + 1, c_2], \ldots, [c_t, b]$ are said to *cover* $[a, b]$. We note that a border interval is covered by at most $d - 1$ canonical intervals. This, together with Fact 2, implies the following

Fact 3 For every interval $[a, b] \subseteq Dom$, there exists a set of $\leq 2(d-1)$ canonical intervals covering $[a, b]$.

We note that results similar to Fact 3 have already been studied in other papers (see, e.g., [19]), but we could not find range query protocols based on them with provable privacy properties.

The RQ protocol π_2. We would like to construct $\pi_2 = (\mathsf{Init}_2, \mathsf{Query}_2)$ as an extension of $\pi_1 = (\mathsf{Init}_1, \mathsf{Query}_1)$, based on the above notions of canonical intervals, and interval covering.

At initialization S again splits database D into two databases: a rank database rD and a payload database pD. This time, however, since we consider exponential-size value domains (as opposed to linear-size value domains used for π_1), S cannot store at initialization the lower rank and the upper rank of each domain value in rD. Then, instead of storing all domain elements in rD, we store all attribute values $u_1, \ldots, u_n$ in D and, for each interval $[u_{i-1} + 1, u_i - 1]$, we consider the set of canonical intervals covering it, as guaranteed by Fact 3, and store each one of these intervals in rD. Note that in each of these latter intervals, each domain value has the same lower and upper ranks, so we only need to store a single copy of these two values in rD as well. Thus, in $rD = (rD_1, rD_2)$, the column rD_1 contains the following

1. the attribute values $u_1, \ldots, u_n$ in D;
2. each one of the canonical intervals covering every interval $[u_{i-1} + 1, u_i - 1]$, where $u_1, \ldots, u_n$ are the attribute values in D.

After this modification, the remaining computations in Init_2, including of the lower/upper ranks, continue as in π_1. Because of Fact 3, this modified initialization at most increases the size of rD by a multiplicative factor of $2(d - 1)$.

At query time, denoting as $[v_0, v_1]$ the range queried by C, the computation of the shifted lower/upper ranks continue as in Query_1. However, C not only asks S for the lower rank of v_0 and the upper rank of v_1 by 2 KS queries as in π_1, but also makes KS queries on input all canonical intervals that contain v_0 and all canonical intervals that contain v_1. Here, note that if v_0 (resp., v_1) is different

from all attribute values, then exactly one of the canonical intervals containing v_0 (resp., v_1) was included in rD during initialization. Thus, only one of the KS queries associated with v_0 and only one of the KS queries associated with v_1 will be successfully completed, returning to C ranks for either an attribute value u_i or a canonical interval containing the query range value. From now on, protocol Query_2 continues exactly as Query_1. That is, C can use the obtained ranks to generate the $m(qr)$ keyword queries to database pD, and obtain $m(qr)$ matching records.

Properties of π_2. The proofs that π_2 satisfies the correctness, privacy and efficiency properties defined in the 2-party model are obtained by extending the analogue proofs for π_1, using the properties of the KS protocol π_0. In particular, the correctness property of π_2 is showed by additionally using Fact 2 and Fact 3. The privacy and the communication complexity properties are not significantly affected by the modifications in π_2 with respect to π_1. The S-time complexity changes by observing that rD is larger in π_2 by a multiplicative factor of d.

4 Range Queries in the Three-Party Model

We show a 3-party RQ protocol by extending the 2-party protocol in Section 3.1. Our protocol follows the general structure outlined in Figure 2 and satisfies the following

Theorem 3. Consider a database with n records and domain $Dom = [0, n-1]$. Assuming the existence of a pseudo-random function, there exists (constructively) a 3-party privacy-preserving RQ protocol $\pi_3 = (\mathsf{Init}_3, \mathsf{Query}_3)$ for such a database, satisfying:

1. correctness
2. privacy against C (i.e., it only leaks the matching records to C);
3. privacy against S (i.e., it does not leak anything to S);
4. privacy against TP (i.e., it only leaks number of matching records, the repetition of query values and the repeated access to initialization encrypted data structures);
5. communication complexity of Query_3 on a queried range qr is $O(m(qr))$;
6. the TP-time complexity in Query_3 on a queried range qr is $O(m(qr) \log n)$.

Remark: on exponential-size domains. We stated Theorem 3 for linear-size value domains, and established it by transforming the 2-party protocol π_1 into the 3-party model. By a very similar transformation, we can adapt the 2-party protocol π_2 into the 3-party model, and obtain a similar result for exponential-size value domains.

Our RQ protocol in the 3-party model: an informal description. Briefly speaking, our protocol π_3 is obtained by performing the following two main modifications in the 3-party model to protocol π_1 (which was designed in the 2-party model): (1) the KS protocol in the 2-party model is replaced by a KS

protocol in the 3-party model [7], that was constructed starting from any pseudo-random function; and (2) the shifts performed to the entire databases rD_2 and pD_1 in protocol π_1 are now replaced by a 'lazy shifting' technique, according to which shifts are performed only to database entries which are used in the protocol. We note that the first modification replaces the use of asymmetric cryptography protocols with only symmetric cryptography techniques, and the second modification eliminates linear-time computations from S during the query subprotocol.

In fact, we can use the following simplified version of the KS protocol in the 3-party model from [7], by assuming that each keyword query will have at most 1 matching record (which was shown to be the case in π_1). First of all, S encrypts both the attribute column and the payload column in its database, where the attribute column is encrypted using deterministic encryption, via a pseudo-random permutation (which can be built from any pseudo-random function). As a result, the encrypted attribute column is searchable by TP using a conventional search data structure (i.e., a binary search tree). Later, S sends the encrypted database to TP and C sends its query values encrypted using the same pseudo-random permutation used by S (with key unknown to TP). Finally, TP can search such value in the search data structure over the encrypted attribute values and return the matching record to C.

Given the above 3-party KS protocol, our 3-party RQ protocol π_3 works as follows. The following high-level structure of π_1 remains in π_3: specifically, S constructs a rank database rD and a payload database pD, and C will perform keyword queries first based on rD and later based on pD. In π_3, however, S sends encrypted versions of rD and pD to TP, and from then on, C only performs keyword queries to TP. Specifically, while the payload columns of rD and pD are encrypted using conventional probabilistic encryption, the attribute columns of rD and pD are encrypted using deterministic encryption, based on a pseudo-random permutation, which makes attribute column values searchable by TP. To encrypt an attribute value $v \in Dom$, S randomly chooses v_0 and an *initial shift* s such that $v_0 + is = v \mathrm{mod} n$, and returns ciphertext $(f_k(v_0), is)$, where f is the pseudo-random permutation, and k is a key known to C and S but not to TP. An interesting property of such ciphertexts is that TP can compute a 'lazy shift' of v over its encryption and by any random *next shift* ns, by returning $(f_k(v_0), cs)$, where the *current shift* cs is $= is + ns \mathrm{mod} n$. Such ciphertexts will be used by S to encrypt lower and upper ranks in rD before sending them to TP. Then, after C's keyword query to (the encrypted version of) database rD held by TP, such encrypted ranks will not be directly returned to C (or otherwise this may leak some information to C across multiple queries). Instead, TP and C will run a 2-party secure function evaluation (using Yao's protocol [25]), where C provides key k as input, TP provides the encrypted ranks and the current shift cs as input and the output returned to TP will be the encrypted queries for (the encrypted version of) database pD. Then, by using the current shift as input to the secure function evaluation protocol, TP obtains encrypted keyword queries, each of them being used to search across the first ciphertext component

of all encrypted attribute values in pD, exactly as done in the above 3-party KS protocol.

Practical performance of our 3-party protocol. In our implementation, the S and TP processes and an instance of MySQL server version 5.5.28 were running on a Dell PowerEdge R710 server with two Intel Xeon X5650 2.66Ghz processors, 48GB of memory, 64-bit Ubuntu 12.04.1 operating system, and connected to a Dell PowerVault MD1200 disk array with 12 2TB 7.2K RPM SAS RQives in RAID6 configuration. The C process was running on a Dell PowerEdge R810 server with two Intel Xeon E7-4870 2.40GHz processors, 64 GB of memory, 64-bit Red Hat Enterprise Linux Server release 6.3 operating system, and connected to the Dell PowerEdge R710 server via switched Gigabit Ethernet.

The 3-party protocol that we implemented was somewhat different than the ones discussed in this paper, because it was developed under more complex and specific project requirements. However, by protocol analysis, we have noted that these differences are not expected to significantly affect practical performance of the protocols. Accordingly, we briefly report on the performance of our implemented protocols, as a useful indication on the performance of the protocols described here.

In our implementation, we have noted practical efficiency and scalability of our 3-party protocols, and were able to achieve query latency performance of no more than 1 order of magnitude slower than a comparable non-private protocol for the same task (specifically, a mySQL protocol for range queries over same-size value domains and database size). This result was achieved, with minor differences, over both linear-size and exponential-size value domains. A similar performance result was presented in [7] in the same implementation environment for keyword queries. In achieving such a result for range queries, our approach of constructing range query protocols from keyword query protocols was critical. This is especially the case when considering that the dominating performance factor in all our range query protocols is given by the performance of one keyword query for each of the records matching the range query. Performance numbers (where time is measured in milliseconds and communication in bytes) for range queries matching 1% of the database records, are captured in Figures 3 and 4.

The most challenging aspect in our performance analysis was the scalability of the initialization procedure, where we observed the following results: the initialization of the 3-party protocol for linear-size value domains, based on a transformation of the 2-party protocol π_1, does achieve satisfactory scalability properties; however, the initalization phase of the 3-party protocol for exponential-size value domains, based on a transformation of the 2-party protocol π_2, does not achieve satisfactory scalability properties, especially as the logarithm of the domain size grows. Although the initialization procedure is typically a one-time procedure, we still consider the following an interesting open problem: designing a 3-party privacy-preserving range query protocol that achieves scalable performance on both query latency and initialization.

Acknowledgements. Many thanks to Euthimios Panagos and Aditya Naidu for helping on performance evaluation. Most of this work was supported by the

DB Size	Bytes Sent	Bytes Received	Response Time
10k	12,927	48,891	978.00
100k	99,327	486,291	1,445.75
500k	483,327	2,430,303	6,712.50
1M	963,327	4,860,321	13,066.25

Fig. 3. Time and communication performance for different database sizes.

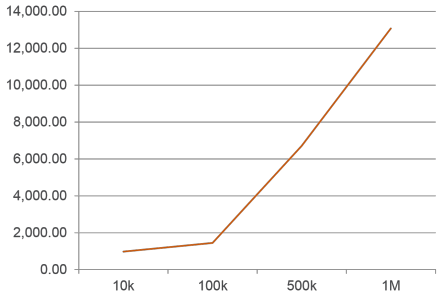

Fig. 4. Time performance as a function of database size.

Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior National Business Center (DoI/NBC) contract number D13PC00003. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, or the U.S. Government.

References

1. R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Order-preserving encryption for numeric data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 563–574, 2004.
2. A. Boldyreva, N. Chenette, and A. O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 578–595, 2011.
3. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.
4. D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 535–554, 2007.
5. B. Chor, N. Gilboa, and M. Naor. Private information retrieval by keywords. *IACR Cryptology ePrint Archive*, 1998.
6. B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.
7. G. Di Crescenzo, D. Cook, A. McIntosh, and E. Panagos. Practical private information retrieval from a time-varying, multi-attribute, and multiple-occurrence database. In *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, pages 339–355, 2014.

8. G. Di Crescenzo, Y. Ishai, and R. Ostrovsky. Universal service-providers for private information retrieval. *J. Cryptology*, 14(1):37–74, 2001.
9. G. Di Crescenzo and D. Shallcross. On minimizing the size of encrypted databases, In *DBSec*, 2014.
10. M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *TCC*, pages 303–324, 2005.
11. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
12. O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.
13. H. Hacigümüs, B. R. Iyer, C. Li, and S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *SIGMOD Conference*, pages 216–227, 2002.
14. B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, Toronto, Canada, August 31 - September 3 2004*, pages 720–731, 2004.
15. M. S. Islam, M. Kuzu, and M. Kantarcioglu. Access pattern disclosure on searchable encryption: Ramification, attack and mitigation. In *NDSS*, 2012.
16. M. S. Islam, M. Kuzu, and M. Kantarcioglu. Inference attack against encrypted range queries on outsourced databases. In *Fourth ACM Conference on Data and Application Security and Privacy, CODASPY'14, San Antonio, TX, USA - March 03 - 05, 2014*, pages 235–246, 2014.
17. S. Jarecki, C. S. Jutla, H. Krawczyk, M.-C. Rosu, and M. Steiner. Outsourced symmetric private information retrieval. In *ACM Conference on Computer and Communications Security*, pages 875–888, 2013.
18. E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *FOCS*, pages 364–373, 1997.
19. J. Li and E. Omiecinski. Efficiency and security trade-off in supporting range queries on encrypted databases. In *Data and Applications Security XIX, 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, USA, August 7-10, 2005, Proceedings*, pages 69–83, 2005.
20. R. Ostrovsky and W. Skeith. A survey of single-database private information retrieval: Techniques and applications. In *Public Key Cryptography*, pages 393–411, 2007.
21. P. Samarati and S. De Capitani di Vimercati. Data protection in outsourcing scenarios: issues and directions. In *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010*, pages 1–14, 2010.
22. E. Shi, J. Bethencourt, H. T. Chan, D. X. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pages 350–364, 2007.
23. E. Stefanov, M. van Dijk, E. Shi, C. W. Fletcher, L. Ren, X. Yu, and S. Devadas. Path ORAM: an extremely simple oblivious RAM protocol. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 299–310, 2013.
24. S. Wang, X. Ding, R. H. Deng, and F. Bao. Private information retrieval using trusted hardware. In *ESORICS*, pages 49–64, 2006.
25. A. C.-C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.