Information Sharing Across Private Databases

Rakesh Agrawal Alexandre Evfimievski * Ramakrishnan Srikant

IBM Almaden Research Center
650 Harry Road, San Jose, CA 95120

ABSTRACT

Literature on information integration across databases tacitly assumes that the data in each database can be revealed to the other databases. However, there is an increasing need for sharing information across autonomous entities in such a way that no information apart from the answer to the query is revealed. We formalize the notion of minimal information sharing across private databases, and develop protocols for intersection, equijoin, intersection size, and equijoin size. We also show how new applications can be built using the proposed protocols.

1. INTRODUCTION

Information integration has long been an area of active database research [12, 16, 21, 27, 48]. So far, this literature has tacitly assumed that the information in each database can be freely shared. However, there is now an increasing need for computing queries across databases belonging to autonomous entities in such a way that no more information than necessary is revealed from each database to the other databases. This need is driven by several trends:

- **End-to-end Integration**: E-business on demand requires end-to-end integration of information systems, from the supply chain to the customer-facing systems. This integration occurs across autonomous enterprises, so full disclosure of information in each database is undesirable.
- **Outsourcing**: Enterprises are outsourcing tasks that are not part of their core competency. They need to integrate their database systems for purposes such as inventory control.
- **Simultaneously compete and cooperate**: It is becoming common for enterprises to cooperate in certain areas and compete in others, which requires selective information sharing.
- **Security**: Government agencies need to share information for devising effective security measures, both within the same government and across governments. However, an agency cannot indiscriminately open up its database to all other agencies.
- **Privacy**: Privacy legislation and stated privacy policies place limits on information sharing. However, it is still desirable to mine across databases while respecting privacy limits.

We propose a new paradigm of **minimal necessary information sharing** across private databases. Intuitively, given a database query spanning multiple private databases, we wish to compute the answer to the query without revealing any additional information apart from the query result. We will sometimes relax this constraint to allow some minimal additional information to be revealed.

1.1 Motivating Applications

We give two prototypical applications to make the above paradigm concrete.

Application 1: Selective Document Sharing Enterprise R is shopping for technology and wishes to find out if enterprise S has some intellectual property it might want to license. However, R would not like to reveal its complete technology shopping list, nor would S like to reveal all its unpublished intellectual property. Rather, they would like to first find the specific technologies for which there is a match, and then reveal information only about those technologies. This problem can be abstracted as follows.

We have two databases D_R and D_S, where each database contains a set of documents. The documents have been preprocessed to only include the most significant words, using some measure such as term frequency times inverse document frequency [41]. We wish to find all pairs of similar documents $d_R \in D_R$ and $d_S \in D_S$, without revealing the other documents. In database terminology, we want to compute the join of D_R and D_S using the join predicate $f(|d_R \cap d_S|, |d_R|, |d_S|) > \tau$, for some similarity function f and threshold τ. The function f could be $|d_R \cap d_S|/(|d_R| + |d_S|)$, for instance.

Many applications map to this abstraction. For example, two government agencies may want to share documents, but only on a need-to-know basis. They would like to find similar documents contained in their repositories in order to initiate their exchange.

Application 2: Medical Research Imagine a future where many people have their DNA sequenced. A medical researcher wants to validate a hypothesis connecting a DNA sequence D with a reaction to drug G. People who have taken the drug are partitioned into four groups, based on whether or not they had an adverse reaction and whether or not their DNA contained the specific sequence; the researcher needs the number of people in each group. DNA sequences and medical histories are stored in databases in autonomous enterprises. Due to privacy concerns, the enterprises do not wish to provide any information about an individual's DNA sequence or medical history, but still wish to help with the research.

*Currently at Cornell University, Ithaca, NY 14853.

SIGMOD 2003, June 9-12, 2003, San Diego, CA.

Assume that the table T_R(person_id, pattern) stores whether a person's DNA contains pattern D and T_S(person_id, drug, reaction) captures whether a person took drug G and whether the person had an adverse reaction. T_R and T_S belong to two different enterprises. The researcher wants to get the answer to the following query.

```
select pattern, reaction, count(*)
from T_R, T_S
where T_R.person_id = T_S.person_id and T_S.drug = "true"
group by T_R.pattern, T_S.reaction
```

We want the property that the researcher should get to know the counts and nothing else, and the enterprises should not learn any new information about any individual.

1.2 Current Techniques

We discuss next some existing techniques that one might use for building the above applications, and why they are inadequate.

- **Trusted Third Party**: The main parties give the data to a "trusted" third party and have the third party do the computation [7, 30]. However, the third party has to be *completely* trusted, both with respect to intent and competence against security breaches. The level of trust required is too high for this solution to be acceptable.
- **Secure Multi-Party Computation**: Given two parties with inputs x and y respectively, the goal of secure multi-party computation is to compute a function $f(x, y)$ such that the two parties learn only $f(x, y)$, and nothing else. See [26, 34] for a discussion of various approaches to this problem.

  Yao [49] showed that any multi-party computation can be solved by building a combinatorial circuit, and simulating that circuit. A variant of Yao's protocol is presented in [37] where the number of oblivious transfers is proportional to the number of inputs and not the size of the circuit. We show in Appendix A that our specialized algorithms are substantially faster than using a circuit, and in particular, the communication costs for circuits make them impractical for our problems.

1.3 Paper Outline

The rest of the paper is organized as follows. We formally state the problem and the scope of this paper in Section 2. We develop the protocol for computing the intersection of two sets in Section 3, and extend this protocol for equijoins in Section 4. We describe the protocols for intersection size and equijoin size in Section 5. In Section 6, we give a cost analysis of these protocols, and use this analysis to estimate the execution times of the application examples above. We conclude with a summary and directions for future work in Section 7.

2. MINIMAL INFORMATION SHARING

2.1 Security Model

We develop our solutions in a setting in which there is no third party [26]. The main parties directly execute a protocol, which is designed to guarantee that they do not learn any more than they would have learnt had they given the data to a trusted third party and got back the answer.

We assume *honest-but-curious* behavior [26]. The parties follow the protocol properly with the exception that they may keep a record of all the intermediate computations and received messages, and analyze the messages to try to learn additional information.

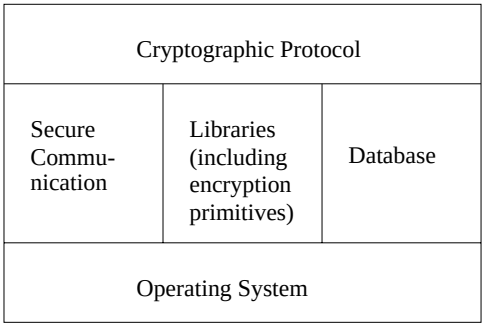

Figure 1: System Components

This behavior is also referred to as *semi-honest* or *passive* behavior.

Figure 1 shows the different components required for building a system for information integration with minimal sharing. Our focus will be on the cryptographic protocol. We assume the use of standard libraries or packages for secure communication and encryption primitives.

2.2 Problem Statement

We now formally state the problem we study in this paper.

Problem Statement (Ideal) Let there be two parties R (receiver) and S (sender) with databases D_R and D_S respectively. Given a database query Q spanning the tables in D_R and D_S, compute the answer to Q and return it to R without revealing any additional information to either party. □

Problem Statement (Minimal Sharing) Let there be two parties R and S with databases D_R and D_S respectively. Given a database query Q spanning the tables in D_R and D_S, and some categories of information I, compute the answer to Q and return it to R without revealing any additional information to either party except for information contained in I. □

For example, if the query Q is a join $T_R \bowtie T_S$ over two tables T_R and T_S, the additional information I might be the number of records in each table: $|T_R|$ and $|T_S|$. Note that whatever R can infer from knowing the answer to the query Q and the additional information I is fair game. For instance, if the query Q is an intersection $V_S \cap V_R$ between two sets V_S and V_R, then for all $v \in (V_R - (V_S \cap V_R))$, R knows that these values were not in V_S.

We assume that the query Q is revealed to both parties. One can think of other applications where the format of Q is revealed, but not the parameters of Q (e.g., in private information retrieval, discussed in Section 2.4).

2.2.1 Operations

In this paper, we focus on four operations: intersection, equijoin, intersection size, and equijoin size.

Let S have a database table T_S, and R have a table T_R, with both tables having a specific attribute A in their schemas. The attribute A takes its values from a given set V. Let V_S be the set of values (without duplicates) that occur in $T_S.A$, and let V_R be the set of values occurring in $T_R.A$. For each $v \in V_S$, let $\mathrm{ext}(v)$ be all records in T_S where $T_S.A = v$, i.e., $\mathrm{ext}(v)$ is the *extra information* in T_S pertaining to v. We show how to compute three kinds of queries over T_S and T_R:

- *Intersection:* Party R learns the set $V_S \cap V_R$, the value $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 3).

- *Equijoin:* Party R learns $V_S \cap V_R$, $\mathtt{ext}(v)$ for all $v \in V_S \cap V_R$, $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 4).
- *Intersection Size:* Party R learns the values of $|V_S \cap V_R|$, $|V_S|$, and nothing else; party S learns $|V_R|$ and nothing else (Section 5).

Thus in the terminology of our problem statement above, the query Q for the three problems corresponds to $V_S \cap V_R$, $T_S \bowtie T_R$ (with $\mathtt{ext}(v)$ used to compute the join), and $|V_S \cap V_R|$ respectively. In all three cases, the additional information I consists of $|V_R|$ and $|V_S|$.

We also extend the intersection size protocol to obtain an *equijoin size* protocol that computes $|T_S \bowtie T_R|$ (Section 5.2). However, R learns $|V_S|$, the distribution of duplicates in $T_S.A$, and based on the distribution of duplicates, some subset of information in $V_S \cap V_R$. S learns $|V_R|$ and the distribution of duplicates in $T_R.A$.

2.3 Limitations

Multiple Queries While we provide guarantees on how much the parties learn from a single query, our techniques do not address the question of what the parties might learn by combining the results of multiple queries. The first line of defence against this problem is the scrutiny of the queries by the parties. In addition, query restriction techniques from the statistical database literature [1, 44] can also help. These techniques include restricting the size of query results [17, 23], controlling the overlap among successive queries [19], and keeping audit trails of all answered queries to detect possible compromises [13].

Schema Discovery and Heterogeneity We do not address the question of how to find which database contains which tables and what the attribute names are; we assume that the database schemas are known. We also do not address issues of schema heterogeneity. See [21, 29] and references therein for some approaches to these problems.

2.4 Related Work

In [35], the authors consider the problem of finding the intersection of two lists while revealing only the intersection. They present two solutions: the first involves oblivious evaluations of n polynomials of degree n each, where n is the number of elements in the list; the second solution requires oblivious evaluation of n^2 linear polynomials. In the context of databases, n will be quite large. In [28], the authors consider the problem of finding people with common preferences, without revealing the preferences. They give intersection protocols that are similar to ours, but do not provide proofs of security.

In the problem of private information retrieval [11, 14, 15, 32, 45], the receiver R obtains the ith record from set of n records held by the sender S without revealing i to S. With the additional restriction that R should only learn the value of one record, the problem becomes that of symmetric private information retrieval [25]. This literature will be useful for developing protocols for the selection operation in our setting.

The problem of privacy-preserving data mining is also related. The randomization approach [6, 22, 40] focuses on individual privacy rather than on database privacy, and reveals randomized information about each record in exchange for not having to reveal the original records to anyone. More closely related is the work in [33] on building a decision-tree classifier across multiple databases, without revealing the individual records in each database to the other databases. Algorithms for mining associations rules across multiple databases have been described in [31] and [47] for horizontally and vertically partitioned data respectively.

The context for the work presented in this paper is our effort to design information systems that protect the privacy and ownership of individual information while not impeding the flow of information. Our other related papers include [2, 3, 4, 5].

3. INTERSECTION

3.1 A Simple, but Incorrect, Protocol

A straightforward idea for computing the intersection $V_S \cap V_R$ would be to use one-way hash functions [38]. Here is a simple protocol that *appears* to work:

1. Both S and R apply hash function h to their sets, yielding $X_S = h(V_S) = \{h(v) \mid v \in V_S\}$ and $X_R = h(V_R) = \{h(v) \mid v \in V_R\}$.
2. S sends its hashed set X_S to R.
3. R sets aside all $v \in V_R$ for which $h(v) \in X_S$; these values form the set $V_S \cap V_R$.

Unfortunately, R can learn a lot more about V_S (with honest-but-curious behavior). For any arbitrary value $v \in V - (V_S \cap V_R)$, R can simply compute $h(v)$ and check whether $h(v) \in X_S$ to determine whether or not $v \in V_S$. In fact, if the domain V is small, R can exhaustively go over all possible values and completely learn V_S.

The intersection protocol we propose next fixes the deficiencies of this protocol.

3.2 Building Blocks

We first describe two building blocks used in the proposed protocols.

3.2.1 Commutative Encryption

Our definition of commutative encryption below is similar to the constructions used in [9, 18, 20, 42] and others. Informally, a commutative encryption is a pair of encryption functions f and g such that $f(g(v)) = g(f(v))$. Thus by using the combination $f(g(v))$ to encrypt v, we can ensure that R cannot compute the encryption of a value without the help of S. In addition, even though the encryption is a combination of two functions, each party can apply their function first and still get the same result.

DEFINITION 1 (INDISTINGUISHABILITY). *Let $\Omega_k \subseteq \{0,1\}^k$ be a finite domain of k-bit numbers. Let $\mathcal{D}_1 = \mathcal{D}_1(\Omega_k)$ and $\mathcal{D}_2 = \mathcal{D}_2(\Omega_k)$ be distributions over Ω_k. Let $\mathcal{A}_k(x)$ be an algorithm that, given $x \in \Omega_k$, returns either true or false. We define distribution $\mathcal{D}_1$ of random variable $x \in \Omega_k$ to be* computationally indistinguishable *from distribution $\mathcal{D}_2$ if for any family of polynomial-step (w.r.t. k) algorithms $\mathcal{A}_k(x)$, any polynomial $p(k)$, and all sufficiently large k*

$$\Pr[\mathcal{A}_k(x) \mid x \sim \mathcal{D}_1] - \Pr[\mathcal{A}_k(x) \mid x \sim \mathcal{D}_2] < \frac{1}{p(k)}$$

where $x \sim \mathcal{D}$ denotes that x is distributed according to $\mathcal{D}$, and $\Pr[\mathcal{A}_k(x)]$ is the probability that $\mathcal{A}_k(x)$ returns true.

Throughout this paper, we will use "indistinguishable" as shorthand for "computationally indistinguishable".

DEFINITION 2 (COMMUTATIVE ENCRYPTION). *A* commutative encryption *$\mathcal{F}$ is a computable (in polynomial time) function $f : \mathrm{Key}\,\mathcal{F} \times \mathrm{Dom}\,\mathcal{F} \to \mathrm{Dom}\,\mathcal{F}$, defined on finite computable domains, that satisfies all properties listed below. We denote $f_e(x) \equiv f(e,x)$, and use "$\in_r$" to mean "is chosen uniformly at random from".*

1. *Commutativity: For all* $e, e' \in \mathrm{Key}\ \mathcal{F}$ *we have*

$$f_e \circ f_{e'} = f_{e'} \circ f_e\ .$$

2. *Each* $f_e : \mathrm{Dom}\ \mathcal{F} \to \mathrm{Dom}\ \mathcal{F}$ *is a bijection.*
3. *The inverse* f_e^{-1} *is also computable in polynomial time given* e.
4. *The distribution of* $\langle x, f_e(x), y, f_e(y)\rangle$ *is indistinguishable from the distribution of* $\langle x, f_e(x), y, z\rangle$, *where* $x, y, z \in_r \mathrm{Dom}\ \mathcal{F}$ *and* $e \in_r \mathrm{Key}\ \mathcal{F}$.

Informally, Property 1 says that when we compositely encrypt with two different keys, the result is the same irrespective of the order of encryption. Property 2 says that two different values will never have the same encrypted value. Property 3 says that given an encrypted value $f_e(x)$ and the encryption key e, we can find x in polynomial time.[1] Property 4 says that given a value x and its encryption $f_e(x)$ (but not the key e), for a new value y, we cannot distinguish between $f_e(y)$ and a random value z in polynomial time. Thus we can neither encrypt y nor decrypt $f_e(y)$ in polynomial time. Note that this property holds only if x is a random value from $\mathrm{Dom}\ \mathcal{F}$, i.e., the adversary does not control the choice of x.

Example 1 Let $\mathrm{Dom}\ \mathcal{F}$ be all quadratic residues modulo p, where p is a "safe" prime number, i.e., both p and $q = (p-1)/2$ are primes. Let $\mathrm{Key}\ \mathcal{F}$ be $\{1, 2, \ldots, q-1\}$. Then, assuming the Decisional Diffie-Hellman hypothesis (DDH) [10], the power function

$$f_e(x) \equiv x^e \bmod p$$

is a commutative encryption:

- The powers commute: $(x^d \bmod p)^e \bmod p = x^{de} \bmod p = (x^e \bmod p)^d \bmod p$.
- Each of the powers f_e is a bijection with its inverse being $f_e^{-1} = f_{e^{-1} \bmod q}$.
- DDH claims that for any generating ($\neq 1$) element $g \in \mathrm{Dom}\ \mathcal{F}$ the distribution of $\langle g^a, g^b, g^{ab}\rangle$ is indistinguishable from the distribution of $\langle g^a, g^b, g^c\rangle$, where $a, b, c \in_r \mathrm{Key}\ \mathcal{F}$. A 3-tuple $\langle g^a, g^b, z\rangle$ from the DDH can be reduced to our 4-tuple $\langle x, x^e, y, z\rangle$ by taking $d \in_r \mathrm{Key}\ \mathcal{F}$ and making tuple $\langle g^d, (g^a)^d, g^b, z\rangle$. Now a plays the role of e, g^d of x, and g^b of y; we test whether $z = (g^b)^a$ or is random. Thus, given DDH, $\langle x, x^e, y, y^e\rangle$ and $\langle x, x^e, y, z\rangle$ are also indistinguishable.

3.2.2 Hash Function

Besides a commutative encryption $\mathcal{F}$, we need a hash function to encode the values $v \in V$ into $x \in \mathrm{Dom}\ \mathcal{F}$. The hashes of values should not collide and should "look random," i.e., there should be no dependency between them that could help encrypt or decrypt one hashed value given the encryption of another. Since we apply commutative encryption to the hashed values $h(v)$ instead of v, the input for the encryption function will appear random, and we will be able to use Property 4 of commutative encryption to prove that our protocols are secure.

In the proofs of our security statements we shall rely on the standard *random oracle model* [8, 24, 46]. We assume that our hash function $h : V \to \mathrm{Dom}\ \mathcal{F}$ is *ideal*, which means that $h(v)$ can be considered computed by a random oracle: every time $h(v)$ is evaluated for a new $v \in V$, an independent random $x \in_r \mathrm{Dom}\ \mathcal{F}$ is chosen for $x = h(v)$.

[1] We only need this property for the join protocol, not for the intersection protocol.

We assume also that $|\mathrm{Dom}\ \mathcal{F}|$ is so large compared to $|V_S \cup V_R|$ that the probability of a collision is exponentially small. Let $N = |\mathrm{Dom}\ \mathcal{F}|$; in the random oracle model, the probability that n hash values have at least one collision equals [46]:

$$\Pr[\text{collision}] \;=\; 1 - \prod_{i=1}^{n-1} \frac{N-i}{N} \;\approx\; 1 - \exp\left(\frac{-n\,(n-1)}{2N}\right)$$

With 1024-bit hash values, half of which are quadratic residues, we have $N \approx 2^{1024}/2 \approx 10^{307}$, and for $n = 1$ million

$$\Pr[\text{collision}] \;\approx\; 1 - \exp\left(-\frac{10^{12}}{10^{307}}\right) \;\approx\; \frac{10^{12}}{10^{307}} \;=\; 10^{-295}.$$

For real-life hash functions, a collision within V_S or V_R can be detected by the server at the start of each protocol by sorting the hashes. If there is a collision between $v \in V_S$ and $v' \in V_R$, it will cause inclusion of v' into the join (or intersection) by R and the disclosure to R of S's records containing v.[2]

3.3 Intersection Protocol

Our proposed intersection protocol is as follows.

1. Both S and R apply hash function h to their sets:
   $X_S = h(V_S)$ and $X_R = h(V_R)$.
   Each party randomly chooses a secret key:
   $e_S \in_r \mathrm{Key}\ \mathcal{F}$ for S and $e_R \in_r \mathrm{Key}\ \mathcal{F}$ for R.
2. Both parties encrypt their hashed sets:
   $Y_S = f_{e_S}(X_S) = f_{e_S}(h(V_S))$ and
   $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.
3. R sends to S its encrypted set $Y_R = f_{e_R}(h(V_R))$, reordered lexicographically.[3]
4. (a) S ships to R its set $Y_S = f_{e_S}(h(V_S))$, reordered lexicographically.
   (b) S encrypts each $y \in Y_R$ with S's key e_S and sends back to R pairs $\langle y, f_{e_S}(y)\rangle = \langle f_{e_R}(h(v)), f_{e_S}(f_{e_R}(h(v)))\rangle$.
5. R encrypts each $y \in Y_S$ with e_R, obtaining
   $Z_S = f_{e_R}(f_{e_S}(h(V_S)))$.
   Also, from pairs $\langle f_{e_R}(h(v)), f_{e_S}(f_{e_R}(h(v)))\rangle$ obtained in Step 4(b) for $v \in V_R$, it creates pairs $\langle v, f_{e_S}(f_{e_R}(h(v)))\rangle$ by replacing $f_{e_R}(h(v))$ with the corresponding v.
6. R selects all $v \in V_R$ for which $(f_{e_S}(f_{e_R}(h(v))) \in Z_S$; these values form the set $V_S \cap V_R$.

3.4 Proofs of Correctness and Security

STATEMENT 1. *Assuming there are no hash collisions,* S *learns the size* $|V_R|$ *and* R *learns the size* $|V_S|$ *and the set* $V_S \cap V_R$.

PROOF. By definition, f_{e_S} and f_{e_R} commute and are bijective. Assuming that hash function h has no collisions on $V_S \cup V_R$,

$$v \in V_S \cap V_R \;\text{ iff }\; v \in V_R \text{ and } (f_{e_S} \circ f_{e_R})(h(v)) \in Z_S,$$

which means that R does recover the correct set $V_S \cap V_R$. Both parties also learn the sizes $|V_R|$ and $|V_S|$, since $|V_R| = |Y_R|$ and $|V_S| = |Y_S|$. □

[2] For the join protocol (Section 4), R can check whether there was a collision between $v \in V_S$ and $v' \in V_R$ by having S include the value v in $\mathtt{ext}(v)$.

[3] If we did not reorder and instead sent the values in the same order as the values in V_R, significant additional information could be revealed.

Next we prove that, assuming the parties follow the protocol correctly, they learn nothing else about the other's sets. We first show that even given

$$\begin{pmatrix} x_1 & \dots & x_m \\ f_e(x_1) & \dots & f_e(x_m) \end{pmatrix} \quad \text{and} \quad x_{m+1},$$

there is no polynomial-time algorithm that can determine whether or not a value u is in fact $f_e(x_{m+1})$.

LEMMA 1. *For polynomial m, the distribution of the $2 \times m$-tuple*

$$\begin{pmatrix} x_1 & \dots & x_{m-1} & x_m \\ f_e(x_1) & \dots & f_e(x_{m-1}) & f_e(x_m) \end{pmatrix}$$

is indistinguishable from the distribution of the tuple

$$\begin{pmatrix} x_1 & \dots & x_{m-1} & x_m \\ f_e(x_1) & \dots & f_e(x_{m-1}) & z_m \end{pmatrix},$$

where $\forall i:\ x_i \in_r \mathrm{Dom}\,\mathcal{F}$, $z_m \in_r \mathrm{Dom}\,\mathcal{F}$, and $e \in_r \mathrm{Key}\,\mathcal{F}$.

PROOF. Let us denote the distribution of the upper tuple by $\mathcal{D}_m$, and the distribution of the lower tuple by $\mathcal{D}_{m-1}$. If $\mathcal{D}_m$ and $\mathcal{D}_{m-1}$ are distinguishable by some polynomial algorithm $\mathcal{A}$, then $\langle x, f_e(x), y, f_e(y)\rangle$ and $\langle x, f_e(x), y, z\rangle$ from Property 4 of commutative encryption are also distinguishable by the following algorithm that takes $\langle x, f_e(x), y, u\rangle$ as argument:

1. For $i = 1 \dots m-1$, let $x_i = f_{e_i}(x)$ and $z_i = f_{e_i}(f_e(x))$, where $e_i \in_r \mathrm{Key}\,\mathcal{F}$;
2. Let $x_m = y$ and $z_m = u$;
3. Submit tuple

$$\begin{pmatrix} x_1 & \dots & x_m \\ z_1 & \dots & z_m \end{pmatrix}$$

to algorithm $\mathcal{A}$ and output whatever it outputs.

For $i = 1 \dots m-1$, we have

$$z_i = f_{e_i}(f_e(x)) = f_e(f_{e_i}(x)) = f_e(x_i),$$

and all x_i are indistinguishable from uniformly random (from Property 4 of commutative encryption). Therefore the distribution of the tuple given to $\mathcal{A}$ is indistinguishable from $\mathcal{D}_m$ when $\langle x, f_e(x), y, u\rangle$ is distributed as $\langle x, f_e(x), y, f_e(y)\rangle$, and from $\mathcal{D}_{m-1}$ when $\langle x, f_e(x), y, u\rangle$ is distributed as $\langle x, f_e(x), y, z\rangle$. So the assumption that $\mathcal{D}_m$ and $\mathcal{D}_{m-1}$ are distinguishable leads to the contradiction that Property 4 does not hold. □

LEMMA 2. *For polynomial m and n, the distribution of the $2 \times n$-tuple*

$$\begin{pmatrix} x_1 & \dots & x_m & x_{m+1} & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_m) & f_e(x_{m+1}) & \dots & f_e(x_n) \end{pmatrix}$$

is indistinguishable from the distribution of the tuple

$$\begin{pmatrix} x_1 & \dots & x_m & x_{m+1} & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_m) & z_{m+1} & \dots & z_n \end{pmatrix},$$

where $0 \leqslant m \leqslant n$, $\forall i:\ x_i, z_i \in_r \mathrm{Dom}\,\mathcal{F}$, and $e \in_r \mathrm{Key}\,\mathcal{F}$.

PROOF. Let us denote by $\mathcal{D}_m^n$ the distribution of the lower tuple; the upper tuple's distribution is thus $\mathcal{D}_n^n$.

From Lemma 1, for all $j = m+1 \dots n$, the distributions $\mathcal{D}_j^n$ and $\mathcal{D}_{j-1}^n$ are indistinguishable. (The first j columns of $\mathcal{D}_j^n$ are identical to $\mathcal{D}_j$ of Lemma 1, the first j columns of $\mathcal{D}_{j-1}^n$ are identical to $\mathcal{D}_{j-1}$ of Lemma 1, and the last $n-j$ columns of $\mathcal{D}_{j-1}^n$ and $\mathcal{D}_j^n$ are just uniformly random numbers.)

Since $\mathcal{D}_{j-1}^n$ and $\mathcal{D}_j^n$ are indistinguishable for $\forall j = m+1 \dots n$, and because n is bounded by a polynomial, $\mathcal{D}_n^n$ is also indistinguishable from any $\mathcal{D}_m^n$ (where $0 \leqslant m \leqslant n$). Let $\mathcal{A}_k$ be an algorithm that pretends to distinguish $\mathcal{D}_n^n$ from $\mathcal{D}_m^n$, and returns true or false. Now

$$\begin{aligned} &\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_n^n] - \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_m^n] \qquad (1) \\ &= \sum_{j=m+1}^{n} \Big(\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_j^n] - \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_{j-1}^n] \Big) \end{aligned}$$

Here k is the number of bits in the tuple values. Consider any polynomial $p(k)$; we want to prove that $\exists k_0\ \forall k \geqslant k_0$ the difference (1) is bounded by $1/p(k)$. Let $p'(k) = n\,p(k)$, which is also a polynomial. We have $\forall j = m+1 \dots n\ \exists k_j\ \forall k \geqslant k_j$ the j-th difference in the telescoping sum is bounded by $1/p'(k)$. Now set $k_0 = \max_j k_j$, and we are done:

$$\sum_{j=m+1}^{n} \Big(\Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_j^n] - \Pr[\mathcal{A}_k(T) \mid T \sim \mathcal{D}_{j-1}^n] \Big) < \sum_{j=m+1}^{n} \frac{1}{p'(k)} < \frac{n}{n\,p(k)} = \frac{1}{p(k)}.$$

Therefore $\mathcal{D}_n^n$ and $\mathcal{D}_m^n$ are computationally indistinguishable. □

STATEMENT 2. *The intersection protocol is secure if both parties are semi-honest. In the end, S learns only the size $|V_R|$, and R learns only the size $|V_S|$ and the intersection $V_S \cap V_R$.*

PROOF. We use a standard proof methodology from multi-party secure computation [26]. If, for any V_S and V_R, the distribution of the S's view of the protocol (the information S gets from R) cannot be distinguished from a simulation of this view that uses only V_S and $|V_R|$, then clearly S cannot learn anything from the inputs it gets from R except for $|V_R|$. Note that the simulation only uses the knowledge S is supposed to have at the end of the protocol, while the distinguisher also uses the inputs of R (i.e., V_R), but not R's secret keys (i.e., e_R). It is important that the distinguisher be unable to distinguish between the simulation and the real view even given R's inputs: this precludes the kind of attack that broke the protocol given in Section 3.1.

The simulator for S (that simulates what S receives from R) is easy to construct. At Step 3 of the protocol, the only step where S receives anything, the simulator generates $|V_R|$ random values $z_i \in_r \mathrm{Dom}\,\mathcal{F}$ and orders them lexicographically. In the real protocol, these values equal $f_{e_R}(h(v))$ for $v \in V_R$. Assuming that, for all $v \in V_R$, the hashes $h(v)$ are distributed uniformly at random (random oracle model), by Lemma 2 the distributions

$$\underbrace{\begin{pmatrix} x_1 & \dots & x_m \\ f_{e_R}(x_1) & \dots & f_{e_R}(x_m) \end{pmatrix}}_{x_i = h(v_i),\ v_i \in V_R} \quad \text{and} \quad \underbrace{\begin{pmatrix} x_1 & \dots & x_m \\ z_1 & \dots & z_m \end{pmatrix}}_{x_i = h(v_i),\ v_i \in V_R},$$

where $\forall i:\ z_i \in_r \mathrm{Dom}\,\mathcal{F}$, are indistinguishable. Therefore the real and simulated views for S are also indistinguishable.

The simulator for R (that simulates what R gets from S) will use V_R, $V_S \cap V_R$ and $|V_S|$; it also knows the hash function h. However, it does not have $V_S - V_R$. The simulator chooses a key $\hat{e}_S \in_r \mathrm{Key}\,\mathcal{F}$. In Step 4(a), the simulation creates Y_S as follows:

- First, for values $v_i \in V_S \cap V_R$, the simulation adds $f_{\hat{e}_S}(h(v_i))$ to Y_S.
- Next, the simulation adds $|V_S - V_R|$ random values $z_i \in_r \mathrm{Dom}\,\mathcal{F}$ to Y_S.

In Step 4(b), the simulation uses the key $\tilde{e}_S$ to encrypt each $y \in Y_R$.

Since e_S (real view) and $\tilde{e}_S$ (simulation) are both chosen at random, their distributions are identical. According to Lemma 2, one cannot distinguish between the distribution of

$$\left(\begin{array}{cccccc} x_1 & \ldots & x_m & x_{m+1} & \ldots & x_n \\ \underbrace{f_{\tilde{e}_S}(x_1) \quad \ldots \quad f_{\tilde{e}_S}(x_m)}_{x_i = h(v_i),\ v_i \in V_R} & & & \underbrace{f_{\tilde{e}_S}(x_{m+1}) \quad \ldots \quad f_{\tilde{e}_S}(x_n)}_{x_i = h(v_i),\ v_i \in V_S - V_R} & & \end{array} \right)$$

and the distribution of

$$\left(\begin{array}{cccccc} x_1 & \ldots & x_m & x_{m+1} & \ldots & x_n \\ \underbrace{f_{\tilde{e}_S}(x_1) \quad \ldots \quad f_{\tilde{e}_S}(x_m)}_{x_i = h(v_i),\ v_i \in V_R} & & & \underbrace{z_{m+1} \quad \ldots \quad z_n}_{x_i = h(v_i),\ v_i \in V_S - V_R} & & \end{array} \right).$$

The real view corresponds to the upper matrix, and the simulated view to the lower matrix. The only difference is that some variables appear in the view encrypted by f_{e_R}, which makes the view a efficiently-computable function of the matrix. Therefore the real view and the simulated view are also indistinguishable, and the statement is proven. □

4. EQUIJOIN

We now extend the intersection protocol so that, in addition to $V_S \cap V_R$, R learns some extra information $\mathrm{ext}(v)$ from S for values $v \in V_S \cap V_R$, but does not learn $\mathrm{ext}(v)$ for for $v \in V_S - V_R$. To compute the join $T_S \bowtie T_R$ on attribute A, we have $\mathrm{ext}(v)$ contain all the records of S's table where $T_S.A = v$, i.e., $\mathrm{ext}(v)$ contains the information about the other attributes in T_S needed for the join.

4.1 Idea Behind Protocol

A simple, but incorrect, solution would be to encrypt the extra information $\mathrm{ext}(v)$ using $h(v)$ as the encryption key. Since, in our intersection protocol, $h(v)$ could not be discovered by R except for $v \in V_R$ (and similarly for S), one might think that this protocol would be secure. While it is true that $h(v)$ cannot be discovered from Y_R or Y_S, $h(v)$ can be discovered from the encryption of ext(v). For any arbitrary value v, R can compute $h(v)$ and try decrypting all the $\mathrm{ext}(v)$ using $h(v)$ to learn whether or not $v \in V_S$. In fact, if the domain is small, R can exhaustively go over all possible values and completely learn both V_S and $\mathrm{ext}(v)$ for $v \in V_S$.

Rather then encrypt the extra information with $h(v)$, we will encrypt it with a key $\kappa(v) = f_{e'_S}(h(v))$, where e'_S is a second secret key of S. The problem now is to allow R to learn $\kappa(v)$ for $v \in V_R$ without revealing V_R to S. We do this as follows: R sends $f_{e_R}(h(v))$ to S, and S sends back $f_{e'_S}(f_{e_R}(h(v)))$ to R. R can now apply $f^{-1}_{e_R}$ to the latter to get

$$f^{-1}_{e_R}(f_{e'_S}(f_{e_R}(h(v)))) = f^{-1}_{e_R}(f_{e_R}(f_{e'_S}(h(v)))) = f_{e'_S}(h(v)).$$

Note that R only gets $f_{e'_S}(h(v))$ for $v \in V_R$, not for $v \in V_S - V_R$.

4.2 Encryption Function K

We now formally define the encryption function $K(\kappa, \mathrm{ext}(v))$ that encrypts $\mathrm{ext}(v)$ using the key $\kappa(v)$. K is defined to be a function

$$K : \mathrm{Dom}\,\mathcal{F} \times V_{\mathrm{ext}} \rightarrow C_{\mathrm{ext}}$$

with two properties:

1. Each function $K_\kappa(x) \equiv K(\kappa, x)$ can be efficiently inverted (decrypted) given κ;
2. "Perfect Secrecy" [43]: For any $\mathrm{ext}(v)$, the value of $K_\kappa(\mathrm{ext}(v))$ is indistinguishable from a fixed (independent of $\mathrm{ext}(v)$) distribution $\mathcal{D}_{\mathrm{ext}}$ over C_{ext} when $\kappa \in_r \mathrm{Dom}\,\mathcal{F}$.

Example 2 Let $\mathcal{F}$ be the power function over quadratic residues modulo a safe prime, as in Example 1. If the extra information $\mathrm{ext}(v)$ can also be encoded as a quadratic residue (i.e., $V_{\mathrm{ext}} = \mathrm{Dom}\,\mathcal{F}$), the encryption $K_\kappa(\mathrm{ext}(v))$ can be just a multiplication operation:

$$K_\kappa(\mathrm{ext}(v)) = \kappa \cdot \mathrm{ext}(v).$$

The multiplication can be easily reversed given κ, and if κ is uniformly random then $\kappa \cdot \mathrm{ext}(v)$ is also uniformly random (independently of $\mathrm{ext}(v)$).

4.3 Equijoin Protocol

Let V_S be the set of values (without duplicates) that occur in $T_S.A$, and let V_R be the set of values that occur in $T_R.A$. For each $v \in V_S$, let $\mathrm{ext}(v)$ be all records in T_S where $T_S.A = v$.

1. Both S and R apply hash function h to their sets: $X_S = h(V_S)$ and $X_R = h(V_R)$.
   R chooses its secret key $e_R \in_r \mathrm{Key}\,\mathcal{F}$, and S chooses two secret keys: $e_S, e'_S \in_r \mathrm{Key}\,\mathcal{F}$.
2. R encrypts its hashed set: $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.
3. R sends to S its encrypted set Y_R, reordered lexicographically.
4. S encrypts each $y \in Y_R$ with both key e_S and key e'_S, and sends back to R 3-tuples $\langle y,\ f_{e_S}(y),\ f_{e'_S}(y)\rangle$ $= \langle f_{e_R}(h(v)),\ f_{e_S}(f_{e_R}(h(v))),\ f_{e'_S}(f_{e_R}(h(v)))\rangle$.
5. For each $v \in V_S$, S does the following:
   (a) Encrypts the hash $h(v)$ with e_S, obtaining $f_{e_S}(h(v))$.
   (b) Generates the key for extra information using e'_S: $\kappa(v) = f_{e'_S}(h(v))$.
   (c) Encrypts the extra information: $c(v) = K(\kappa(v), \mathrm{ext}(v))$.
   (d) Forms a pair $\langle f_{e_S}(h(v)),\ c(v)\rangle$ $= \langle f_{e_S}(h(v)),\ K(f_{e'_S}(h(v)), \mathrm{ext}(v))\rangle$.

   The pairs are then shipped to R in lexicographical order.
6. R applies $f^{-1}_{e_R}$ to all entries in the 3-tuples received at Step 4, obtaining $\langle h(v), f_{e_S}(h(v)), f_{e'_S}(h(v))\rangle$ for all $v \in V_R$.
7. R sets aside all pairs $\langle f_{e_S}(h(v)), K(f_{e'_S}(h(v)), \mathrm{ext}(v))\rangle$ received at Step 5 whose first entry occurs as a second entry in a 3-tuple $\langle h(v), f_{e_S}(h(v)), f_{e'_S}(h(v))\rangle$ from Step 6. Using the third entry $f_{e'_S}(h(v)) = \kappa(v)$ as the key, R decrypts $K(f_{e'_S}(h(v)), \mathrm{ext}(v))$ and gets $\mathrm{ext}(v)$. The corresponding v's form the intersection $V_S \cap V_R$.
8. R uses $\mathrm{ext}(v)$ for $v \in V_S \cap V_R$ to compute $T_S \bowtie T_R$.

4.4 Proofs of Correctness and Security

STATEMENT 3. *Assuming there are no hash collisions, S learns $|V_R|$, and R learns $|V_S|$, $V_S \cap V_R$, and $\mathrm{ext}(v)$ for all $v \in V_S \cap V_R$.*

PROOF. This protocol is an extension of the intersection protocol, so it allows R to determine $V_S \cap V_R$ correctly. Since R learns the keys $\kappa(v)$ for values in the intersection, R also gets $\mathrm{ext}(v)$ for $v \in V_S \cap V_R$. □

Next we prove that R and S do not learn anything besides the above. We first extend Lemma 2 as follows.

LEMMA 3. *For polynomial n, the distributions of the following two $3\times n$-tuples*

$$\begin{pmatrix} x_1 & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_n) \\ f_{e'}(x_1) & \dots & f_{e'}(x_n) \end{pmatrix} \quad \textit{and} \quad \begin{pmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \\ z_1 & \dots & z_n \end{pmatrix},$$

are computationally indistinguishable, where $\forall i : x_i, y_i, z_i \in_r \mathrm{Dom}\,\mathcal{F}$, and $e, e' \in_r \mathrm{Key}\,\mathcal{F}$.

PROOF. Let us denote the left distribution by $\mathcal{D}_1$, the right distribution by $\mathcal{D}_2$, and the following "intermediate" distribution by $\mathcal{D}_3$:

$$\begin{pmatrix} x_1 & \dots & x_n \\ f_e(x_1) & \dots & f_e(x_n) \\ z_1 & \dots & z_n \end{pmatrix}$$

The first and third line in the tuples for $\mathcal{D}_1$ and $\mathcal{D}_3$ are distributed like $\mathcal{D}_n^n$ and $\mathcal{D}_0^n$ (from Lemma 2) respectively. The second line in both $\mathcal{D}_1$ and $\mathcal{D}_3$ can be obtained from the first line by applying f_e with random key e. Therefore, since $\mathcal{D}_n^n$ and $\mathcal{D}_0^n$ are indistinguishable by Lemma 2, distributions $\mathcal{D}_1$ and $\mathcal{D}_3$ are also indistinguishable.

Analogously, the first and second lines in $\mathcal{D}_3$ and $\mathcal{D}_2$ are distributed like $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ respectively. The third line in both $\mathcal{D}_3$ and $\mathcal{D}_2$ can be obtained by using random numbers for the z_i's. Therefore, by Lemma 2, $\mathcal{D}_3$ and $\mathcal{D}_2$ are also indistinguishable.

Finally, since both $\mathcal{D}_1$ and $\mathcal{D}_2$ are indistinguishable from $\mathcal{D}_3$, they themselves are indistinguishable. □

The following lemma will be used in the proof for the security of the join protocol to show that the real and simulated views for R are indistinguishable. $\mathcal{D}_1'$ corresponds to the real view (for R), while $\mathcal{D}_2'$ corresponds to the simulated view. The first t columns correspond to $V_S - (V_S \cap V_R)$, the next $m-t$ columns to $V_S \cap V_R$, and the last $n-m$ columns to $V_R - (V_S \cap V_R)$.

LEMMA 4. *For polynomial m, t, and n, and any $c_i \in V_{\mathrm{ext}}$, the two distributions $\mathcal{D}_1'$ and $\mathcal{D}_2'$ of the $4\times n$-tuple*

$$\begin{pmatrix} x_1 & \dots & x_t & x_{t+1} & \dots & x_m & x_{m+1} & \dots & x_n \\ y_1 & \dots & y_t & y_{t+1} & \dots & y_m & y_{m+1} & \dots & y_n \\ & & & z_{t+1} & \dots & z_m & z_{m+1} & \dots & z_n \\ \xi_1 & \dots & \xi_t & \xi_{t+1} & \dots & \xi_m & & & \end{pmatrix}$$

such that

- *For $\mathcal{D}_1'$, $\forall i : x_i \in_r \mathrm{Dom}\,\mathcal{F}$, $y_i = f_e(x_i)$, $z_i = f_{e'}(x_i)$, and $\xi_i = K(f_{e'}(x_i), c_i)$ where $e, e' \in_r \mathrm{Key}\,\mathcal{F}$;*
- *For $\mathcal{D}_2'$, $\forall i : x_i, y_i, z_i \in_r \mathrm{Dom}\,\mathcal{F}$, and*
  - *$i = 1 \dots t : \xi_i$ is independent random with distribution $\mathcal{D}_{\mathrm{ext}}$,*
  - *$i = t{+}1 \dots m : \xi_i = K(z_i, c_i)$*

are computationally indistinguishable. (In both $\mathcal{D}_1'$ and $\mathcal{D}_2'$, the positions corresponding to $z_1 \dots z_t$ and $\xi_{m+1} \dots \xi_n$ are blank.)

PROOF. Denote by $\mathcal{D}_3'$ the following "intermediate" distribution:

$$\forall i : x_i, y_i, z_i \in_r \mathrm{Dom}\,\mathcal{F} \text{ and } \xi_i = K(z_i, c_i).$$

Note that the z_i for $i = 1 \dots t$ are not included in the tuple, even though they are used to generate $K(z_i, c_i)$.

The only difference between the two distributions $\mathcal{D}_2'$ and $\mathcal{D}_3'$ is that, for $i = 1 \dots t$, we replace ξ_i distributed as $\mathcal{D}_{\mathrm{ext}}$ with $K(z_i, c_i)$ where $z_i \in_r \mathrm{Dom}\,\mathcal{F}$; the rest of the matrix is independent and stays the same. Since z_i is not a part of the matrix for $i = 1 \dots t$, by Property 2 of encryption $K(\kappa, c)$, distributions $\mathcal{D}_2'$ and $\mathcal{D}_3'$ are indistinguishable.

Next we use Lemma 3 to show that distributions $\mathcal{D}_1'$ and $\mathcal{D}_3'$ are also indistinguishable. We define function $Q(M)$ that takes a $3\times n$ matrix M (from Lemma 3) and generates a $4 \times n$ matrix M' as follows:

1. The first 3 rows of M' are the same as the first 3 rows of M, except that the values corresponding to $z_1, \dots, z_t$ in M' are left blank.
2. The fourth row of M' is generated by taking $\xi_i = K(z_i, c_i)$ where z_i is the corresponding value of the third row of M.

If M is distributed like $\mathcal{D}_1$ of Lemma 3, $Q(M)$ corresponds to $\mathcal{D}_1'$. If M is distributed like $\mathcal{D}_2$, $Q(M)$ corresponds to $\mathcal{D}_3'$. Since by Lemma 3, $\mathcal{D}_1$ and $\mathcal{D}_2$ are indistinguishable, and $Q(M)$ is computable in polynomial time, $\mathcal{D}_1'$ and $\mathcal{D}_3'$ are also indistinguishable.

Finally, since both $\mathcal{D}_1'$ and $\mathcal{D}_2'$ are indistinguishable from $\mathcal{D}_3'$, they themselves are indistinguishable. □

STATEMENT 4. *The join protocol is secure if both parties are semi-honest. At the end of the protocol, S learns only $|V_R|$; R learns only $|V_S|$, $V_S \cap V_R$, and $\mathrm{ext}(v)$ for all $v \in V_S \cap V_R$.*

PROOF. As in the proof of Statement 2, we will construct simulators of each party's view of the protocol, such that each simulator is given only what the party is supposed to learn, and such that the distribution of the real view is indistinguishable from the distribution of the simulated view.

The simulator for S is identical to that in Statement 2, since S gets exactly the same input from R as in the intersection protocol. Hence the proof from Statement 2 directly applies.

The simulator for R (that simulates what R receives from S) can use h, e_R, V_R, $V_S \cap V_R$, $\mathrm{ext}(v)$ for $v \in V_S \cap V_R$, and $|V_S|$. Let

$$V_S = \{v_1, \dots, v_t, v_{t+1}, \dots, v_m\} \text{ and}$$
$$V_R = \{v_{t+1}, \dots, v_m, v_{m+1}, \dots, v_n\}.$$

So $t = |V_S - V_R|$, $m = |V_S|$, and $n = |V_S \cup V_R|$. Note that the simulator does not know the values in $V_S - V_R$.

In Step 4, the simulator generates n random numbers $y_i \in_r \mathrm{Dom}\,\mathcal{F}$, $i = 1 \dots n$ as the simulated values for $f_{e_S}(h(v_i))$, and an additional n random numbers $z_i \in_r \mathrm{Dom}\,\mathcal{F}$ as the simulated values for $f_{e'_S}(h(v_i))$. The simulation then uses key e_R to create

$$\langle f_{e_R}(h(v_i)),\ f_{e_R}(y_i),\ f_{e_R}(z_i)\rangle$$

for $i = t{+}1 \dots m$. These triplets are ordered lexicographically and comprise the simulated view for Step 4.

In Step 5, the simulator creates the pairs as follows:

- For values $v_{t+1}, \dots, v_m$ from $V_S \cap V_R$, the simulator encrypts $\mathrm{ext}(v_i)$ as $\xi_i = K(z_i, \mathrm{ext}(v_i))$; then it forms pairs $\langle y_i, \xi_i\rangle$;
- For $i = 1 \dots t$, the simulator creates $|V_S - V_R|$ additional pairs $\langle y_i, \xi_i\rangle$ where ξ_i have distribution $\mathcal{D}_{\mathrm{ext}}$ over C_{ext}, i.e., y_i and ξ_i are random values from their respective domains.

These pairs are sorted lexicographically and comprise the simulated view for Step 5.

Setting $x_i = h(v_i)$, the real view corresponds to distribution $\mathcal{D}_1'$ of the matrix in Lemma 4, while the simulation corresponds to distribution $\mathcal{D}_2'$ of the matrix. The only difference is that some variables appear in the view encrypted by f_{e_R}, which makes the view a efficiently-computable function of the matrix. Since these $\mathcal{D}_1'$ and $\mathcal{D}_2'$ are indistinguishable, the simulation is also indistinguishable from the real view. □

5. INTERSECTION AND JOIN SIZES

5.1 Intersection Size

We now show how the intersection protocol can be modified, such that R only learns the intersection size, but not which values in V_R were present in V_S. (Simply applying the intersection protocol would reveal the set $V_R \cap V_S$, in addition to the intersection size.) Recall that in Step 4 of the intersection protocol, S sends back to R the values of $y \in Y_R$ together with their encryptions made by S. These encryptions are paired with the unencrypted y's so that R can match the encryptions with R's values. If instead S sends back to R only the lexicographically reordered encryptions of the y's and not the y's themselves, R can no longer do the matching.

5.1.1 Intersection Size Protocol

We now present the protocol for intersection size. (Steps 1 through 3 are the same as in the intersection protocol.)

1. Both S and R apply hash function h to their sets: $X_S = h(V_S)$ and $X_R = h(V_R)$. Each party randomly chooses a secret key: $e_S \in_r \text{Key}\,\mathcal{F}$ for S and $e_R \in_r \text{Key}\,\mathcal{F}$ for R.

2. Both parties encrypt their hashed sets: $Y_S = f_{e_S}(X_S) = f_{e_S}(h(V_S))$ and $Y_R = f_{e_R}(X_R) = f_{e_R}(h(V_R))$.

3. R sends to S its encrypted set $Y_R = f_{e_R}(h(V_R))$, reordered lexicographically.

4. (a) S ships to R its set $Y_S = f_{e_S}(h(V_S))$, reordered lexicographically.
(b) S encrypts each $y \in Y_R$ with S's key e_S and sends back to R the set $Z_R = f_{e_S}(Y_R) = f_{e_S}(f_{e_R}(h(V_R)))$, reordered lexicographically.

5. R encrypts each $y \in Y_S$ with e_R, obtaining $Z_S = f_{e_R}(f_{e_S}(h(V_S)))$.

6. Finally, R computes intersection size $|Z_S \cap Z_R|$, which equals $|V_S \cap V_R|$.

5.1.2 Proofs of Correctness and Security

STATEMENT 5. *Assuming there are no hash collisions, S learns the size $|V_R|$ and R learns the size $|V_S|$ and the size $|V_S \cap V_R|$.*

PROOF. The proof is very similar to that for Statement 1. Since f_{e_S} and f_{e_R} commute and are bijective, assuming that hash function h has no collisions on $V_S \cup V_R$,

$$|V_S \cap V_R| \;=\; f_{e_R}(f_{e_S}(h(V_S))) \cap f_{e_S}(f_{e_R}(h(V_R))).$$

Therefore R recovers the correct size $|V_S \cap V_R|$. □

STATEMENT 6. *The intersection size protocol is secure if both parties are semi-honest. At the end of the protocol, S learns only the size $|V_R|$, and R learns only the sizes $|V_S|$ and $|V_S \cap V_R|$.*

PROOF. We use the same methodology as in the proofs of Statement 2 and 4.

The simulator for S's view of the intersection size protocol is identical to that in Statement 2, since S gets exactly the same input from R as in the intersection protocol. Hence the proof from Statement 2 directly applies.

The simulator for R's view of the protocol is allowed to use V_R, the hash function h, e_R, and the numbers $|V_S \cap V_R|$ and $|V_S|$; however, it has neither $V_S - V_R$ nor $V_S \cap V_R$. Let

$$V_S \;=\; \{v_1, \ldots, v_t, v_{t+1}, \ldots, v_m\} \text{ and}$$
$$V_R \;=\; \{v_{t+1}, \ldots, v_m, v_{m+1}, \ldots, v_n\}.$$

So $t = |V_S - V_R|$, $m = |V_S|$, and $n = |V_S \cup V_R|$.

The simulator generates n random numbers $y_1, \ldots, y_n \in_r \text{Dom}\,\mathcal{F}$ which play the role of $f_{e_S}(h(v))$ for all $v \in V_S \cup V_R$. The key e_S is not simulated, and no decision is made about which y_i stands for which $f_{e_S}(h(v))$. In Step 4(a), the simulation creates Y_S as

$$Y_S \;=\; \{y_1, \ldots, y_m\}.$$

In Step 4(b), the simulation generates Z_R by taking set $\{y_{t+1}, \ldots, y_n\}$ and encoding it with f_{e_R}:

$$Z_R \;=\; \{f_{e_R}(y_{t+1}), \ldots, f_{e_R}(y_n)\}.$$

We now show that the distribution of R's real view in the protocol is computationally indistinguishable from the distribution of R's simulated view.

According to Lemma 2, the distributions $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ of the following matrix M:

$$\begin{pmatrix} x_1 & \ldots & x_n \\ y_1 & \ldots & y_n \end{pmatrix}$$

where

- $\mathcal{D}_0^n$: $\forall i : x_i, y_i \in_r \text{Dom}\,\mathcal{F}$;
- $\mathcal{D}_n^n$: $\forall i : x_i \in_r \text{Dom}\,\mathcal{F}, y_i = f_{e_S}(x_i), e_S \in_r \text{Key}\,\mathcal{F}$;

are indistinguishable. Given $x_i = h(v_i)$, consider the following function $Q(M)$:

$$Q(M) \;=\; \langle h,\, e_R,\, Y_S,\, Z_R \rangle,$$

where

$$\begin{aligned} h &:= \text{a function on } V_S \cup V_R \text{ s.t. } \forall i :\; h(v_i) = x_i; \\ e_R &:= \text{a random key}; \\ Y_S &:= \{y_1, \ldots, y_m\}; \\ Z_R &:= \{f_{e_R}(y_{t+1}), \ldots, f_{e_R}(y_n)\}. \end{aligned}$$

If M is distributed according to $\mathcal{D}_0^n$, then $Q(M)$ corresponds to the simulated view of server R. If M's distribution is $\mathcal{D}_n^n$, then

$$\begin{aligned} y_i \;&=\; f_{e_S}(x_i) = f_{e_S}(h(v_i)), \\ f_{e_R}(y_i) \;&=\; f_{e_R}(f_{e_S}(x_i)) \;=\; f_{e_S}(\, f_{e_R}(h(v_i))\,), \end{aligned}$$

and $Q(M)$ is distributed like the real view of R. Since from Lemma 2, $\mathcal{D}_0^n$ and $\mathcal{D}_n^n$ are indistinguishable, and Q is computable in polynomial time, the simulated view $Q(\mathcal{D}_0^n)$ and the real view $Q(\mathcal{D}_n^n)$ are also indistinguishable. □

5.2 Equijoin Size

To evaluate equijoin size, we follow the intersection size protocol, except that we allow V_R and V_S to be multi-sets, i.e., contain duplicates, and then compute the join size instead of the intersection size in Step 6. However, R can now use the number of duplicates of a given value to partially match values in Y_R with their corresponding encryptions in Z_R. We now characterize exactly what R and S learn in this protocol (besides $|V_R|$, $|V_S|$ and $|V_R \bowtie V_S|$).

To start with, R learns the distribution of duplicates in V_S, and S learns the distribution of duplicates in V_R. To characterize what else R learns, let us partition the values in V_R based on the number of duplicates, i.e., in a partition $V_R(d)$, each $v \in V_R(d)$ has

d duplicates. Then, for each partition, R learns $|V_R(d) \cap V_S(d')|$ for each partition $V_S(d')$ of V_S. Thus if all values have the same number of duplicates (e.g., no duplicates as in our intersection protocol), R only learns $|V_R \cap V_S|$. At the other extreme, if no two values have the same number of duplicates, R will learn $V_R \cap V_S$.

6. COST ANALYSIS

6.1 Protocols

Let

- each encrypted codeword (in $\mathrm{Dom}\,\mathcal{F}$) be k bits long,
- C_h denote the cost of evaluating the hash function,
- C_e denote the cost of encryption/decryption by $\mathcal{F}$ (e.g., exponentiation "$x^y \bmod p$" over k-bit integers),
- C_K denote the cost of encryption/decryption by K (e.g., encoding/decoding as a quadratic residue and multiplication), and
- $n \log n \cdot C_s$ be the cost of sorting a set of n encryptions.

We assume the obvious optimizations when computing the computation and communication costs. For example, in the join protocol, we assume that the protocol does not decrypt y to $h(v)$ in Step 6, but uses order preservation for matching. Also, in all the protocols, S does not retransmit the y's back but just preserves the original order.

Computation The computation costs are:

- *Intersection:* $(C_h + 2C_e)(|V_S| + |V_R|) + 2C_s|V_S|\log|V_S| + 3C_s|V_R|\log|V_R|$
- *Join:* $C_h(|V_S| + |V_R|) + 2C_e|V_S| + 5C_e|V_R| + C_K(|V_S| + |V_S \cap V_R|) + 2C_s|V_S|\log|V_S| + 3C_s|V_R|\log|V_R|$

We can assume $C_e \gg C_h$, $C_e \gg C_K$, and $nC_e \gg n\log n \cdot C_s$, so these formulae can be approximated by

- *Intersection:* $2C_e(|V_S| + |V_R|)$
- *Join:* $2C_e|V_S| + 5C_e|V_R|$

Communication The communication cost is:

- *Intersection:* $(|V_S| + 2|V_R|) \cdot k$ bits
- *Join:* $(|V_S| + 3|V_R|) \cdot k + |V_S| \cdot k'$ bits, where k' is the size of the encrypted $\mathrm{ext}(v)$.

Both the intersection size and join size protocols have the same computation and communication complexity as the intersection protocol.

6.2 Applications

We now estimate the execution times for the applications in Section 1.1.

For the cost of C_e (i.e., cost of $x^y \bmod p$), we use the times from [36]: 0.02s for 1024-bit numbers on a Pentium III (in 2001). This corresponds to around 2×10^5 exponentiations per hour. We assume that communication is via a T1 line, with bandwidth of 1.544 Mbits/second ($\approx$ 5 Gbits/hour).

Encrypting the set of values is trivially parallelizable in all three protocols. We assume that we have P processors that we can utilize in parallel: we will use a default value of $P = 10$.

V_R := ids in T_R.
V'_R := subset of V_R that match the DNA sequence.
V_S := ids in T_S that took the drug.
V'_S := subset of V_S with adverse reaction.
T gets IntersectionSize(V'_R, V'_S).
T gets IntersectionSize(V'_R, $(V_S - V'_S)$).
T gets IntersectionSize($(V_R - V'_R)$, V'_S).
T gets IntersectionSize($(V_R - V'_R)$, $(V_S - V'_S)$).

Figure 2: Algorithm for Medical Research Application

6.2.1 Selective Document Sharing

Recall that we have two databases D_R and D_S, where each database contains a set of documents, and a document consists of a set of significant words. We wish to find all pairs of documents $d_R \in D_R$ and $d_S \in D_S$ such that, for some similarity function f and threshold τ, $f(|d_R \cap d_S|, |d_R|, |d_S|) > \tau$. For example, f could be $|d_R \cap d_S|/(|d_R| + |d_S|)$.

Implementation R and S execute the intersection size protocol for each pair of documents $d_R \in D_R$ and $d_S \in D_S$ to get $|d_R \cap d_S|$, $|d_R|$ and $|d_S|$; they then compute the similarity function f.

For S, in addition to the number of documents $|D_S|$, this protocol also reveals to R for each document $d_R \in D_R$, which documents in D_S matched d_R, and the size of $|d_R \cap d_S|$ for each document $d_S \in D_S$.

Cost Analysis For a given pair of documents d_R and d_S, the computation time is $(|d_R| + |d_S|) \cdot 2C_e$, and the data transferred is $(|d_R| + 2|d_S|) \cdot k$ bits. Thus the total cost is:

- Computation: $|D_R| \cdot |D_S| \cdot (|d_R| + |d_S|) \cdot 2C_e$.
- Communication: $|D_R| \cdot |D_S| \cdot (|d_R| + 2|d_S|) \cdot k$.

If $|D_R| = 10$ documents, $D_S = 100$ documents, and $|d_R| = |d_S| = 1000$ words, the computation time will be $4 \times 10^6\, C_e/P \approx$ 2 hour. The data transferred will be $3 \times 10^6\, k \approx 3$ Gbits ≈ 35 minutes.

6.2.2 Medical Research

Recall that we wish to get the answer to the query

```
select pattern, reaction, count(*)
from T_R, T_S
where T_R.id = T_S.id and T_S.drug = true
group by T_R.pattern, T_S.reaction
```

where T_R and T_S are tables in two different enterprises.

Implementation Figure 2 shows the implementation algorithm. We use a slightly modified version of the intersection size protocol where Z_R and Z_S are sent to T, the researcher, instead of to S and R. Note that whenever we have, say, $(V_R - V'_R)$ inside IntersectionSize, the set difference is computed locally, and the result is the input to the protocol.

Cost Analysis The combined cost of the four intersections is $2(|V_R| + |V_S|) \cdot 2C_e$, and the data transferred is $2(|V_R| + |V_S|) \cdot 2k$ bits. If $|V_R| = |V_S| = 1$ million, the total computation time will be $8 \times 10^6\, C_e/P \approx 4$ hours. The total communication time will be $8 \times 10^6\, k \approx 8$ Gbits ≈ 1.5 hours.

7. CONCLUSIONS

We identified information integration with minimal sharing as a new area for future database research. We developed novel protocols for three key operations: intersection, intersection size, and

equijoin and proved that these protocols disclose minimal information apart from the query result. We also gave a protocol for computing equijoin size, but this protocol leaks some information about which tuples joined, based on the distribution of duplicates. We also showed how new applications can be built using the proposed protocols.

Some interesting directions for future research include:

- What is the tradeoff between the additional information being disclosed and efficiency? Will we be able to obtain much faster protocols if we are willing to disclose additional information?
- Can we formalize models of minimal disclosure and discover corresponding protocols for other database operations such as aggregations?

Acknowledgments We thank Robert Morris for suggesting motivating applications. We also thank Dalit Naor for pointing out related work in secure multi-party computation.

8. REFERENCES

[1] N. R. Adam and J. C. Wortman. Security-control methods for statistical databases. *ACM Computing Surveys*, 21(4):515–556, Dec. 1989.

[2] R. Agrawal and J. Kiernan. Watermarking relational databases. In *28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[3] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of the 28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Implementing P3P using database technology. In *Proc. of the 19th Int'l Conference on Data Engineering*, Bangalore, India, March 2003.

[5] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. An XPath-based preference language for P3P. In *Proc. of the 12th Int'l World Wide Web Conference*, Budapest, Hungary, May 2003.

[6] R. Agrawal and R. Srikant. Privacy preserving data mining. In *ACM SIGMOD Conference on Management of Data*, pages 439–450, Dallas, Texas, May 2000.

[7] S. Ajmani, R. Morris, and B. Liskov. A trusted third-party computation service. Technical Report MIT-LCS-TR-847, MIT, May 2001.

[8] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *Proc. of the First ACM Conf. on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, 1993.

[9] J. C. Benaloh and M. de Mare. One-way accumulators: A decentralized alternative to digital sinatures. In *Advances in Cryptology – EUROCRYPT'93, Workshop on the Theory and Application of Cryptographic Techniques*, volume 765 of *Lecture Notes in Computer Science*, pages 274–285, Lofthus, Norway, May 1993. Springer-Verlag.

[10] D. Boneh. The decision diffie-hellman problem. In *Proc. of the 3rd International Algorithmic Number Theory Symposium*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63, Portland, Oregon, USA, June 1998. Springer-Verlag.

[11] C. Cachin, S. Micali, and M. Stadler. Computationally private information retrieval with polylogarithmic communication. In *Theory and Application of Cryptographic Techniques*, pages 402–414, 1999.

[12] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[13] F. Chin and G. Ozsoyoglu. Auditing and infrence control in statistical databases. *IEEE Transactions on Software Eng.*, SE-8(6):113–139, April 1982.

[14] B. Chor and N. Gilboa. Computationally private information retrieval. In *Proc. of 29th ACM Symposium on Theory of Computing*, pages 304–313, 1997.

[15] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[16] U. Dayal and H.-Y. Hwang. View definition and generalization for database integration in a multidatabase system. *IEEE Transactions on Software Eng.*, 10(6):628–645, 1984.

[17] D. Denning, P. Denning, and M. Schwartz. The tracker: A threat to statistical database security. *ACM Transactions on Database Systems*, 4(1):76–96, March 1979.

[18] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[19] D. Dobkin, A. Jones, and R. Lipton. Secure databases: Protection against user influence. *ACM Transactions on Database Systems*, 4(1):97–106, March 1979.

[20] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, July 1985.

[21] A. Elmagarmid, M. Rusinkiewicz, and A. Sheth, editors. *Management of Heterogeneous & Autonomous Database Systems*. Morgan Kaufmann, 1999.

[22] A. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proc. of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

[23] I. Fellegi. On the question of statistical confidentiality. *Journal of the American Statistical Assoc.*, 67(337):7–18, March 1972.

[24] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology, Proceedings of Crypto 86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987.

[25] Y. Gertner, Y. Ishai, E. Kushilevitz, and T. Malkin. Protecting data privacy in private information retrieval schemes. In *ACM Symposium on Theory of Computing*, pages 151–160, 1998.

[26] O. Goldreich. Secure multi-party computation. Working Draft, Version 1.3, June 2001.

[27] L. M. Haas, R. J. Miller, B. Niswonger, M. T. Roth, P. M. Schwarz, and E. L. Wimmers. Transforming heterogeneous data with database middleware: Beyond integration. *IEEE Data Engineering Bulletin*, 22(1), 1999.

[28] B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *Proc. of the 1st ACM Conference on Electronic Commerce*, pages 78–86, Denver, Colorado, November 1999.

[29] P. Ipeirotis and L. Gravano. Distributed search over the hidden web: Hierarchical database sampling and selection. In *28th Int'l Conference on Very Large Databases*, Hong Kong, China, August 2002.

[30] N. Jefferies, C. Mitchell, and M. Walker. A proposed architecture for trusted third party services. In *Cryptography Pol-*

icy and Algorithms Conference. Springer LNCS v 1029 pp 98–104, July 1995.

[31] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, June 2002.

[32] E. Kushilevitz and R. Ostrovsky. Replication is not needed: Single database, computationally-private information retrieval. In *Proc. of the IEEE Symp. on Found. of Computer Science (FOCS)*, 1997.

[33] Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

[34] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In *Proc. of the ACM Symposium on Theory of Computing*, pages 590–599, 2001.

[35] M. Naor and B. Pinkas. Oblivious transfer and polynomial evaluation. In *Proc. of the 31th ACM Symposium on Theory of Computing*, pages 245–254, Atlanta, Georgia, 1999.

[36] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proc. of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 448–457, Washington DC, USA, January 2001.

[37] M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proc. of the 1st ACM Conference on Electronic Commerce*, pages 129–139, Denver, Colorado, November 1999.

[38] B. Preneel. Analysis and design of cryptographic hash functions. Ph.D. dissertation, Katholieke Universiteit Leuven, 1992.

[39] M. O. Rabin. How to exchange secrets by oblivious transfer. Technical Memo TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[40] S. J. Rizvi and J. R. Haritsa. Privacy-preserving association rule mining. In *Proc. of the 28th Int'l Conference on Very Large Databases*, August 2002.

[41] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[42] A. Shamir, R. L. Rivest, and L. M. Adleman. Mental poker. Technical Memo MIT-LCS-TM-125, Laboratory for Computer Science, MIT, February 1979.

[43] C. E. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28-4:656–715, 1949.

[44] A. Shoshani. Statistical databases: Characteristics, problems and some solutions. In *Proc. of the Eighth Int'l Conference on Very Large Databases*, pages 208–213, Mexico City, Mexico, September 1982.

[45] S. W. Smith and D. Safford. Practical private information retrieval with secure coprocessors. Research Report RC 21806, IBM, July 2000.

[46] D. R. Stinson. *Cryptography: Theory and Practice*, chapter 4. Chapman & Hall/CRC, second edition, 2002.

[47] J. Vaidya and C. W. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proc. of the 8th ACM SIGKDD Int'l Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.

[48] G. Wiederhold. Intelligent integration of information. In *Proc. of the ACM SIGMOD Conf. on Management of Data*, Washington, DC, 1993.

[49] A. C. Yao. How to generate and exchange secrets. In *Proc. of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Canada, October 1986.

APPENDIX

A. CIRCUIT-BASED PROTOCOLS

For comparison, we estimate the computation and communication cost of intersection and join protocols obtained using the semi-honest variant of Yao's protocol described in [33, 37]. Let V_S and V_R contain w-bit values. Consider a function $f(\vec{x}, \vec{y})$ that takes vectors $\vec{x}$ and $\vec{y}$ (of size $w \cdot |V_S|$ and $w \cdot |V_R|$ respectively) as inputs and returns a vector $\vec{z}$ (of size $|V_R|$) that shows which of R's values also belong to V_S. This function can be represented by a circuit of boolean gates. S hardwires its input $\vec{x}$ into the circuit and supplements each possible encrypted bit value at each circuit wire with its own random key (used for decrypting the next gate's output and its key). The protocol has two major steps:

Coding R's input: For each bit of $\vec{y}$, R engages with S in a 1-out-of-2 oblivious transfer protocol [36, 39] and gets the corresponding supplemental keys.

Computing the circuit: For each gate, R receives a table from S and, using the keys for the gate's inputs, computes the output and its key. In the process, R applies a pseudorandom function twice per each output wire.

To get $f(\vec{x}, \vec{y})$, R gets the tables that allow it to decrypt the wires with the output of the circuit.

A.1 Cost Analysis

Let the keys (for the circuit gates) be k_0 bits long, and C_r be the cost of pseudorandom function evaluation. We assume that $w = 32$ (recall that w is the size in bits of the input values), $k_0 = 64$, and $|V_R| = |V_S| = n$.

A.1.1 Coding the Input

Let C_{ot} be the computation cost of each oblivious transfer, and C'_{ot} its communication cost. An efficient protocol for oblivious transfers is given in [36]. For any integer $l > 0$, this paper gives a protocol with amortized cost

$$C_{\text{ot}} = \frac{1}{l} \cdot C_e + \frac{2^l}{l} \cdot C_{\times}; \quad C'_{\text{ot}} > \frac{2^l}{l} \cdot k_1$$

where $C_{\times}$ is the cost of multiplication, and k_1 is the size of the keys used in oblivious transfer. We assume $k_1 = 100$ [36]. Assume that $C_e = 1000\, C_{\times}$; then the best choice with respect to the computation time is $l = 8$, and the costs become

$$C_{\text{ot}} = 0.157\, C_e; \quad C'_{\text{ot}} > 32\, k_1\,.$$

Cost The computation cost of coding the input is

$$w \cdot |V_R| \cdot C_{\text{ot}} = 32 \times n \times 0.157\, C_e \approx 5\, n\, C_e$$

and the communication cost is

$$w \cdot |V_R| \cdot C'_{\text{ot}} > 32 \times n \times 32\, k_1 \approx 10^5\, n$$

A.1.2 Evaluating the Circuit

Let $C(w, |V_S|, |V_R|)$ be the total number of gates required for the circuit. We estimate lower bounds on the number of gates required for a brute force algorithm, and a more efficient partitioning algorithm.

Let G_e be the number of gates required to compare two w-bit numbers in the circuit to determine whether they are equal. Let G_l be the number of gates required to determine which number is less than (or equal to) the other.

Brute Force Circuit Consider a circuit that compares every number in V_R with every number in V_S, and then merges the results to

output just the numbers in V_R that were equal to at least one number in V_S. The number of gates $C(w, |V_S|, |V_R|)$ in this circuit is greater than

$$|V_R| \cdot |V_S| \cdot G_e.$$

Partitioning Circuit We assume that each set V_R and V_S is given to the circuit in the form of an ordered array, with all duplicates removed. Instead of comparing all pairs of numbers, we can split these arrays into m intervals (non-interleaving subarrays) of size $|V_R|/m$ and $|V_S|/m$. For ease of exposition, we assume that $|V_R| = |V_S| = n$, and that n is a power of m.

Out of all possible m^2 pairs of subarrays, with one subarray from V_S and the other from V_R, only at most $2m - 1$ pairs may have a nonempty intersection; the others are pairs of non-interleaving subarrays. To see this, note that in a pair of interleaving subarrays the beginning of one subarray must be within the interval spanned by the other. There is at most one pair per one such "internal beginning." There are $2m$ subarrays in both V_S and V_R, each having only one beginning; and the smallest beginning is always "wasted," thus limiting the number of interleaving pairs to $2m - 1$.

The circuit has to choose the $2m - 1$ interleaving pairs of subarrays and then use recursion to compute set intersections within these pairs. To check whether a pair of subarrays interleaves, we need to compare the smallest and largest numbers of these subarrays, thus making 2 comparisons. There are m^2 pairs, so we need $2m^2$ comparisons, and hence $2m^2 G_l$ gates. Additional gates are needed to reroute the subarrays and combine the recursive outputs, but we shall ignore them in our estimation, since we are interested primarily in a lower bound for the cost of the circuit (using this algorithm).

Let $f(n)$ be the cost of the circuit. Then

$$\begin{aligned} f(n) &> 2m^2 G_l + (2m-1) \cdot f(n/m) \\ f(1) &= G_e \end{aligned}$$

Let $c = 2m^2 G_l$; then

$$\begin{aligned} f(n) &> c + (2m-1) \cdot f(n/m) \\ &> c + (2m-1) \cdot (c + (2m-1) f(n/m^2)) \\ &> c \cdot \sum_{i=0}^{\log_m n - 1} (2m-1)^i + (2m-1)^{\log_m n} f(1) \\ &= c \cdot \frac{(2m-1)^{\log_m n} - 1}{(2m-1) - 1} + (2m-1)^{\log_m n} G_e \\ &= c \cdot \frac{n^{\log_m (2m-1)} - 1}{2m-2} + n^{\log_m (2m-1)} G_e \\ &> \left(\frac{c}{2m-2} + G_e \right) \cdot (n^{\log_m (2m-1)} - 1) \end{aligned}$$

Substituting back the value for c, we get

$$f(n) > \left(\frac{m^2}{m-1} G_l + G_e \right) \cdot (n^{\log_m (2m-1)} - 1)$$

Two w-bit numbers can be checked for equality using $2w-1$ binary gates and compared using $5w - 3$ gates. Setting $G_l = 5w - 3$ and $G_e = 2w - 1$ gives

$$f(n) > \left(\frac{m^2}{m-1} (5w-3) + (2w-1) \right) \cdot (n^{\log_m (2m-1)} - 1)$$

Brute Force vs. Partitioning Let $|V_R| = |V_S| = n$. (As before w = 32 and k_0 = 64.) Then, for the partitioning circuit, we get the following values for $C(w, |V_R|, |V_S|) = f(n)$ for the optimal value of m:

n	m	$f(n)$
10,000	11	2.3×10^8
1 million	19	7.3×10^{10}
100 million	32	1.9×10^{13}

The brute force circuit does much worse, with $C(w, |V_R|, |V_S|)$ equal to 6.3×10^9, 6.3×10^{13}, and 6.3×10^{17} respectively.

Cost For each gate in the circuit, R gets a table from S whose size is $4k_0$, and evaluates 2 pseudorandom functions. Therefore the computation cost of circuit evaluation is

$$2C_r \cdot C(w, |V_S|, |V_R|) = 2C_r f(n)$$

and the communication cost is

$$4k_0 \cdot C(w, |V_S|, |V_R|) = 256 \cdot f(n).$$

A.2 Comparison with Our Protocol

Computation We get the following computation costs:

n	Circuit		Our Protocol
	Input (OT)	Evaluation	
10^4	$5 \times 10^4 C_e$	$4.7 \times 10^8 C_r$	$4 \times 10^4 C_e$
10^6	$5 \times 10^6 C_e$	$1.5 \times 10^{11} C_r$	$4 \times 10^6 C_e$
10^8	$5 \times 10^8 C_e$	$3.8 \times 10^{13} C_r$	$4 \times 10^8 C_e$

The cost of coding the input for the circuit is slightly higher than the cost of our protocol. The total cost of the circuit (relative to our protocol) depends on the ratio of C_e to C_r. While $C_e \gg C_r$, there are 10^4 to 10^5 as many calls to C_r as there are to C_e. Thus our protocol will be substantially faster if $C_r > C_e/10000$, and slightly faster otherwise.

Communication The communication costs (in bits) are:

n	Circuit		Our Protocol
	Input (OT)	Circuit (Tables)	
10^4	10^9	6.0×10^{10}	3×10^7
10^6	10^{11}	1.8×10^{13}	3×10^9
10^8	10^{13}	4.9×10^{15}	3×10^{11}

The circuit-based protocol requires 1000 to 10,000 times as much communication as our protocol. For n = 1 million, the communication time for the circuit-based protocol is 144 days (using a T1 line), versus 0.5 hours for our protocol. The communication cost makes the circuit-based protocol impractical for database-size applications.