Private Record Matching Using Differential Privacy

Ali Inan
University of Texas at Dallas
Richardson, TX 75083, USA
inan@student.utdallas.edu

Murat Kantarcioglu
University of Texas at Dallas
Richardson, TX 75083, USA
muratk@utdallas.edu

Gabriel Ghinita
Purdue University
W. Lafayette, IN 47907, USA
gghinita@cs.purdue.edu

Elisa Bertino
Purdue University
W. Lafayette, IN 47907, USA
bertino@cs.purdue.edu

ABSTRACT

Private matching between datasets owned by distinct parties is a challenging problem with several applications. Private matching allows two parties to identify the records that are close to each other according to some distance functions, such that no additional information other than the join result is disclosed to any party. Private matching can be solved securely and accurately using secure multi-party computation (SMC) techniques, but such an approach is prohibitively expensive in practice. Previous work proposed the release of sanitized versions of the sensitive datasets which allows *blocking*, i.e., filtering out sub-sets of records that cannot be part of the join result. This way, SMC is applied only to a small fraction of record pairs, reducing the matching cost to acceptable levels. The blocking step is essential for the privacy, accuracy and efficiency of matching. However, the state-of-the-art focuses on sanitization based on k-anonymity, which does not provide sufficient privacy. We propose an alternative design centered on *differential privacy*, a novel paradigm that provides strong privacy guarantees. The realization of the new model presents difficult challenges, such as the evaluation of distance-based matching conditions with the help of only a statistical queries interface. Specialized versions of data indexing structures (e.g., kd-trees) also need to be devised, in order to comply with differential privacy. Experiments conducted on the real-world *Census-income* dataset show that, although our methods provide strong privacy, their effectiveness in reducing matching cost is not far from that of k-anonymity based counterparts.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*; H.2.8 [**Database Management**]: Database applications—*Statistical databases*; H.2.m [**Database Management**]: Miscellaneous

General Terms

Security, Experimentation, Performance

EDBT 2010, March 22–26, 2010, Lausanne, Switzerland.

Keywords

Privacy, Security, Record matching, Differential privacy

1. INTRODUCTION

Analysis and integration of information maintained by distinct entities is critical for a large class of applications. For instance, two competitor businesses may wish to share information about customers with similar demographics (e.g., age, zipcode), if doing so allows them to increase their revenues (e.g., to jointly support a location-customized service for young subscribers). However, to protect their customer base both parties want to keep data that is not part of the join result private. In this scenario, the objective is to match similar records that represent *distinct* individuals, therefore matching based on unique identifiers (e.g., SSN) is not applicable. Another setting that has received significant attention is the *record linkage* problem, which attempts to match records stored at distinct parties (e.g., hospitals), but which represent the *same* entity (e.g., patient). While it is acceptable for hospitals to share information about a patient that they both have treated, it may be a privacy violation to disclose data of other patients.

Private linkage is a challenging problem, and previous research [11] has shown that, in many situations, uniquely-identifying information may not be available, and linkage is performed based on matching of other information, such as age, occupation, etc. To complicate things further, such information may not always be completely consistent across datasets (e.g., the weight of a patient may vary between two admissions to different hospitals). Therefore, it is important to devise methods that are capable of privately linking records through a *distance-based* condition, rather than simple equi-joins computed using cryptographic hashes [2].

Two main approaches have been proposed for private matching: sanitization methods and Secure Multi-party Computation (SMC) protocols [14]. Methods in the former category perturb private information to prevent re-identification of individual records [23, 3, 21]. Typically, data attributes are suppressed, generalized, or transformed in some other way to protect privacy. Sanitization methods have two limitations: (i) the amount of privacy provided may not be sufficient and (ii) the matching result may exhibit false positives/negatives. On the other hand, SMC methods provide strong privacy and perfect accuracy, but incur prohibitive computational and communication cost. All existing SMC techniques that we are aware of [2, 7, 24, 15, 13, 18, 1] require $O(n * m)$ cryptographic operations where n (resp. m) is the number of records in the first (resp. second) data set. If n=m=10000, such an integration task will require 10^8 cryptographic operations. The cost of each individual operation is very high, and grows with the number of compared attributes. Even with cryptographic accelerators, such a protocol is still not usable for practical purposes [17].

The work in [16] has identified the benefits of a hybrid approach that combines sanitization and SMC. The main idea is to identify from the sanitized information those record pairs that cannot possibly satisfy the join condition. This filtering process is called *blocking*. Record pairs that are not eliminated by blocking are included in the SMC step, that can securely and accurately compute the join result. However, the input to the SMC protocol may be considerably smaller (up to 2 orders of magnitude) compared to the pure-SMC approaches. Therefore, the hybrid technique can be used in practical applications. Nevertheless, the approach in [16] sanitizes data according to the k-anonymity paradigm, which may not be sufficient to protect privacy [21]. Furthermore, as proven in [8] for the general case, subsequent privacy formulations that improve on k-anonymity, such as ℓ-diversity [21] or t-closeness [20] may still be vulnerable against adversaries with certain types of background-knowledge.

Recently, the *differential privacy* [8] paradigm emerged as an alternative to data sanitization techniques. Instead of publishing perturbed versions of the dataset, differential privacy allows users to interact with the database only by means of statistical queries. Random noise is added to each query result to preserve data privacy. The work in [8, 9] explores differential privacy in a thorough theoretical framework, and shows that the privacy guarantees provided are very strong. Specifically, an adversary that attempts to attack the privacy of some individual entity r will not be able to distinguish from the interaction with the database (called a *transcript*) whether a record representing r is present or not in the database.

Our objective is to design a blocking protocol that operates within the differential privacy framework. Specifically, each participant discloses only the perturbed results of a set of statistical queries. Devising a differentially private blocking step is a challenging task, due to the following reasons:

- The matching operation requires evaluations of distance - based conditions, whereas differential privacy allows statistical queries only.
- The amount of noise added by the statistical database to protect data is dependent on the *sensitivity* characteristic of the answered queries (the sensitivity concept is formally defined in Section 2.1). The matching protocols must be carefully designed such that the sensitivity of the combination of queries that are answered is kept low.
- Existing data-partitioning index structures, which are necessary to improve blocking effectiveness, disclose a lot of information about data distribution, and are not compliant with differential privacy. Specialized variations of such structures must be carefully designed to ensure privacy.

In this paper, we develop a blocking protocol that supports efficient private matching, and at the same time provides strong data protection compliant with differential privacy. Our specific contributions are:

(*i*) A new security model for private matching in the context of differential privacy.

(*ii*) The design of indexing techniques that are compliant with differential privacy, and a blocking protocol based on the resulting data partitioning.

(*iii*) Several heuristics to improve blocking effectiveness and matching accuracy in the presence of constraints on the maximum number of SMC operations allowed.

(*iv*) An extensive experimental evaluation that shows the effectiveness of the proposed protocol on real data sets.

Section 2 formally describes our security model, and introduces the differential privacy framework. Section 3 gives an overview of our method. Section 4 presents differentially-private mechanisms for data partitioning, based on which we implement the blocking process described in Section 5. Section 6 outlines the SMC post-blocking operations, whereas Section 7 presents the experimental evaluation. We survey related work in Section 8 and conclude with directions for future work in Section 9.

2. BACKGROUND AND DEFINITIONS

Denote by A and B two data holder parties, and by T and V their respective datasets. We assume that both T and V have the same schema[1], with attributes $(A_1, \ldots, A_d)$. Without loss of generality, we also assume that each attribute domain is numerical, and totally ordered. For categorical attributes, this requirement can be enforced by introducing a unique numerical label for each attribute value. We use the interval notation $[x_i, y_i]$ to denote a sub-set of the domain of attribute A_i that includes all values between x_i and y_i (inclusive of the interval boundaries).

The two parties execute a private record matching protocol that identifies similar records in T and V. Record matching is equivalent to building a classifier, i.e., a function $f : T(A_1, \ldots, A_d) \times V(A_1, \ldots, A_d) \rightarrow \{\text{true,false}\}$ that labels record pairs as *"Match"* if the value of f is *true*, or *"Mismatch"* otherwise.

In the private record matching problem, an accurate classifier f is assumed to be available [22]. Therefore the focus of private record matching methods is on applying the classifier to all record pairs within the input datasets privately, accurately and efficiently. Here, we consider distance-based classifiers [10].

We assume that both parties agree beforehand on the characteristics of the classifier, which is expressed as a *decision rule*. Formally,

DEFINITION 1 (DECISION RULE). *Denote by*

$$dist_i : Dom(A_i) \times Dom(A_i) \rightarrow \mathbb{R}^+$$

a distance function defined over the domain of attribute A_i, and by $\theta_i \geq 0$ a matching threshold. Decision rule DR is defined as:

$$DR(t,v) = \left\{ \begin{array}{ll} true & if \forall\, 1 \leq i \leq d, dist_i(t[i], v[i]) \leq \theta_i \\ false & otherwise \end{array} \right\}.$$

A record pair $(t \in T, v \in V)$ *matches* according to DR if $DR(t,v) = true$. Our objective is to identify the join result $T \bowtie_{DR(t,v)} V$ in a privacy preserving manner, such that the result will be available to both data holders A, B and any record of T and V that does not satisfy the join condition is not disclosed. Next, we discuss the privacy model and assumptions of our work. Specifically, in Section 2.1, we outline the differential privacy model [9] that characterizes the maximum amount of information each party is willing to disclose about its data. In Section 2.2 we discuss our privacy assumptions and adversarial model.

2.1 Differential Privacy

A statistical database answers aggregate queries such as $Count$ or Sum queries. In our work, we consider aggregate *range* queries, that can be expressed by d-dimensional hyper-rectangles in the attribute domain space $A_1 \times \ldots \times A_d$. A range query is represented as

[1] If this assumption is not met, the private schema matching protocol from [22] can be employed to obtain a shared schema.

a Cartesian product $Q = [x_1^Q, y_1^Q] \times \ldots \times [x_d^Q, y_d^Q]$ where $[x_i^Q, y_i^Q]$ is the extent of Q on attribute A_i. For instance, the $Count$ query

$$\texttt{SELECT COUNT}(*) \texttt{ FROM T WHERE } (40 \leq Age \leq 60);$$

has extent $[40, 60]$ on attribute Age. Statistical databases allow users to retrieve coarse-grained aggregate information. On the other hand, it is important to preserve the privacy of individual records, by not allowing fine-grained aggregate queries. One solution to preserve privacy is to disallow queries that have extents smaller than a certain threshold, e.g., 10 years of age, and $10k$ salary. However, this may not be sufficient. For instance, consider malicious user Mallory who knows that his colleague Alice is the only one in the company who is 51 years old. Mallory can issue the following two queries:

$$\begin{aligned}
\texttt{Q1}: \quad & \texttt{SELECT COUNT}(*) \texttt{ FROM T} \\
& \texttt{WHERE } (40 \leq Age \leq 50) \texttt{ AND } (30k \leq Salary \leq 40k); \\
\texttt{Q2}: \quad & \texttt{SELECT COUNT}(*) \texttt{ FROM T} \\
& \texttt{WHERE } (40 \leq Age \leq 51) \texttt{ AND } (30k \leq Salary \leq 40k).
\end{aligned}$$

Assume that the response to Q_1 is 10, and the response to Q_2 is 11. Then, Mallory can infer that Alice must be included in the result of query Q_2, and hence her salary is in the range $[30k, 40k]$. Note that, the attack was successful because Mallory was able to access the query results for the two views T_1 and T_2 of table T that differ in a single record (the record of Alice).

Differential privacy [9] aims to prevent this type of inferences, by adding random noise to each query result. The interaction between a user and the database is expressed as a *transcript*. Formally,

DEFINITION 2 (TRANSCRIPT). *Let $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ be a set of aggregate queries, and denote by $Q_i^T\,_{(1 \leq i \leq q)}$ the result of answering query Q_i on table T. A* transcript

$$\mathcal{TR}_{\mathcal{Q}}^T = \{(Q_1, a_1), \ldots, (Q_q, a_q)\}$$

is the response of a statistical database to the query set $\mathcal{Q}$ on database table T, where $a_i = Q_i^T$.

A statistical database satisfies ϵ-differential privacy if the ϵ-indistinguishability condition [9] is fulfilled:

DEFINITION 3 (ϵ-INDISTINGUISHABILITY). *Consider a statistical database that produces the transcript $\mathcal{U}$ on the set of queries $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$, and let $\epsilon > 0$ be an arbitrarily-small real constant. Transcript $\mathcal{U}$ satisfies* ϵ-indistinguishability *if for every pair of views T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, it holds that*

$$\left| \ln \frac{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_1} = \mathcal{U}]}{Pr[\mathcal{TR}_{\mathcal{Q}}^{T_2} = \mathcal{U}]} \right| \leq \epsilon \tag{1}$$

In other words, an attacker is not able to learn whether the transcript was obtained by answering the query set $\mathcal{Q}$ on view T_1, or on view T_2. To achieve ϵ-indistinguishability, a statistical database injects noise into each query result Q_i^T. The amount of noise required is proportional to the *sensitivity* of the query set $\mathcal{Q}$, which is defined as follows:

DEFINITION 4 (L_1-SENSITIVITY). *Over any two views T_1, T_2 such that $|T_1| = |T_2|$ and T_1, T_2 differ in only one record, the L_1*-sensitivity *of query set $\mathcal{Q} = \{Q_1, \ldots, Q_q\}$ is measured as*

$$S_{L_1}(\mathcal{Q}) = \max_{\forall T_1, T_2} \sum_{i=1}^{q} |Q_i^{T_1} - Q_i^{T_2}|$$

The following theorem from [9] gives a sufficient condition for a statistical database to satisfy ϵ-differential privacy:

THEOREM 1. *Let $\mathcal{Q}$ be a set of queries and denote by $S_{L_1}(\mathcal{Q})$ the L_1-sensitivity of $\mathcal{Q}$. Then, differential privacy with parameter ϵ can be achieved by adding to each query result random noise X, i.e., $Q_i^T \leftarrow Q_i^T + X$, where X is a random, i.i.d. variable drawn from a Laplace distribution with mean* 0 *and magnitude $\lambda \geq S_{L_1}(\mathcal{Q})/\epsilon$.*

An essential operation in enforcing differential privacy is determining the sensitivity $S_{L_1}(\mathcal{Q})$. Interestingly, it is shown in [9] that $S_{L_1}(\mathcal{Q})$ is independent of the dataset T, and can be determined based on the query set $\mathcal{Q}$ alone. However, for arbitrary query sets, it is shown in [25] that computing sensitivity is NP-hard. Nevertheless, if certain restrictions are imposed on $\mathcal{Q}$, sensitivity (or its approximation) for sets of $Count$ queries can be efficiently computed. Specifically:

(i) If all queries in $\mathcal{Q}$ have disjoint ranges, $S_{L_1}(\mathcal{Q}) = 2$.

(ii) If queries in $\mathcal{Q}$ have overlapping ranges, then a 2-approximation of $S_{L_1}(\mathcal{Q})$ is given by the maximum number of queries that overlap the same point in the data space. Formally, for any data space point $p \in A_1 \times \ldots \times A_d$, define

$$\mathcal{Q}^p = \{Q \in \mathcal{Q} | p \in Q\}$$

i.e., the set of queries that cover point p. Then, we have

$$S_{L_1}(\mathcal{Q}) \leq 2 \cdot \max_{p \in A_1 \times \ldots \times A_d} |\mathcal{Q}^p|.$$

2.2 Privacy Definition

One of the most important aspects of SMC techniques is that privacy guarantees of the underlying protocols can be proven under reasonable assumptions. We believe that a similar theoretical framework to prove privacy guarantees is needed for our hybrid framework. Not surprisingly, our main goal will be to extend the basic definitions and techniques used in SMC so that they apply to our hybrid framework. Specifically, we will focus on extending security/privacy definitions given for the semi-honest model to our hybrid framework where each party also reveals some sanitized information about its data.[2]

The basic semi-honest behavior definition [14] states that a computation is secure if the view of the dishonest parties (combined view, as dishonest parties may collude) during the execution of the protocol can be effectively simulated knowing only the input and the output of the dishonest parties. This is not quite the same as saying that all private information is protected. Obviously, under this definition, disclosure of any information that can be deduced from the final result is not considered a violation. For example, if two entities that differ in only one attribute are not matched, information about how that attribute affects the matching is revealed. Such inferences cannot be prevented in the context of SMC, as this information can be deduced from the final result and the input.

We extend the basic semi-honest model [14] by including the sanitized data (e.g., statistical query results that satisfy differential privacy definitions) as public data that is accessible by all participants. Formally, let $\bar{a} = (a_1, \ldots, a_m)$ be the sanitized data of all the parties. Let $f : (\{0,1\}^*)^m \longmapsto (\{0,1\}^*)^m$ be a probabilistic, polynomial-time functionality, where $f_i(x_1, x_2, \ldots, x_m)$ denotes the i^{th} component of $f(x_1, x_2, \ldots, x_m)$ and let Π be an m-party

[2] We would like to stress that the semi-honest model is the most commonly used model in SMC-based privacy preserving data analytics algorithms [7].

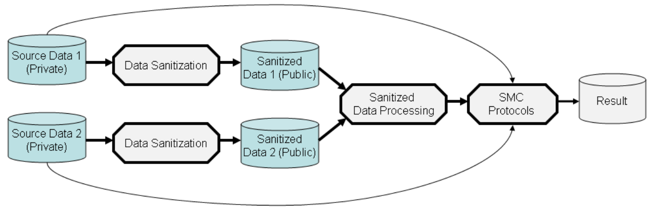


Figure 1: Overview of the hybrid model

protocol for computing f (in the context of this paper, f is the private record matching procedure). For $I = \{i_1, i_2, \dots, i_t\} \subseteq [m]$ where $[m]$ denotes the set $\{1, 2, \dots, m\}$, we let $f_I(x_1, x_2, \dots, x_m)$ denote the sub-sequence $f_{i_1}(x_1, \dots, x_m)$,..., $f_{i_t}(x_1, \dots, x_m)$. Let the view of the i^{th} party during an execution of protocol Π on $\bar{x} = (x_1, x_2, \dots, x_m)$, denoted $view_i^{\Pi}(\bar{x})$, be $\left(x_i, r_i, m_i^1, \dots, m_i^t\right)$ where r_i represents the outcome of the i^{th} party's internal coin tosses, and m_i^j represents the j^{th} message received by the i^{th} party. Also, given $I = \{i_1, i_2, \dots, i_t\}$, we let $view_I^{\Pi}(\bar{x})$ denote the sub-sequence $\left(I, view_{i_1}^{\Pi}(\bar{x}) \dots view_{i_t}^{\Pi}(\bar{x})\right)$.

We say that protocol Π privately computes f if there exists a probabilistic polynomial time algorithm, denoted S, such that for every $I \subseteq [m]$, it holds that

$$\{(S(I, (x_{i_1}, \dots, x_{i_t}), \bar{a}, f_I(\bar{x})), f(\bar{x}))\}_{\bar{x} \in (\{0,1\}^*)^m} \overset{C}{\equiv} \left\{\left(view_I^{\Pi}(\bar{x}, \bar{a}), output^{\Pi}(\bar{x})\right)\right\}_{\bar{x} \in (\{0,1\}^*)^m}$$

where $\overset{C}{\equiv}$ denotes computational indistinguishability [14].

Compared to the existing privacy definitions in the semi-honest model, we assume that all sanitized data (e.g., statistical query results that satisfy differential privacy) are available to any coalition of parties (as shown in Figure 1) and the ultimate goal of the privacy-preserving protocol is to reveal nothing more than what can be inferred by *all* sanitized information, original inputs of the parties that are part of the coalition and the final function result (e.g., the final result of the record linkage procedure).

Compared to classic SMC models, in our hybrid model we can trade off privacy versus efficiency easily. For example, if no sanitized data is revealed, our model will be equivalent to SMC models (e.g., $\bar{a}$ is an empty set in the above definition). On the other hand, as we show in this paper, by revealing the statistical query results that satisfy differential privacy, it is possible to improve the efficiency of the SMC protocols without sacrificing accuracy.

3. OVERVIEW OF THE SOLUTION

Consider two datasets T and V, owned by parties A and B, respectively. Both parties want to learn the result of $T \bowtie_{DR} V$, without disclosing to the other party any records that are not part of the result. In a naive approach, A and B would engage in an SMC protocol that takes as input sets T and V, and privately evaluates the decision rule DR on all record pairs of the Cartesian product $T \times V$. However, such an approach incurs excessive computational and communication costs.

To reduce the private matching cost, A and B partition their data into sub-sets $\{T_i\}_{1 \le i \le m}$ and $\{V_j\}_{1 \le i \le n}$. For each sub-set, the corresponding extent and cardinality are disclosed to the other party. Based on sub-set extents, it can be determined that records from certain sub-set pairs (T_i, V_j) cannot contain matching records, hence such pairs are not included in the SMC phase. The filtering of sub-set pairs is called *blocking*, and can considerably reduce matching overhead.

Note that, if the extent of a sub-set is expressed as its minimum bounding hyper-rectangle, then the information disclosed is equivalent to answering an aggregate $Count$ query with a rectangular range. Therefore, cardinalities of $\{T_i\}_{1 \le i \le m}$ and $\{V_j\}_{1 \le i \le n}$ are equivalent to transcripts in the statistical database model. To protect the privacy of its records, each party controls the amount of information disclosed according to differential privacy for a given privacy parameter ϵ, decided by each party independently[3].

The proposed private matching technique consists of three phases: *(i)* construction of transcripts satisfying differential privacy, *(ii)* sub-set based blocking and *(iii)* SMC-based matching of sub-set pairs that are not filtered during blocking. In Section 3.1, we discuss transcript generation. Next, we show how blocking is performed in Section 3.2. The SMC protocol is presented in detail in Section 6.

3.1 Transcript Generation

The transcript construction executed by each party consists of two steps: partitioning and output perturbation. We illustrate these steps using the record sets T and V shown in Figure 2(a). There are two attributes, Age and $Gender$, and the join decision rule for a pair of records ($t \in T, v \in V$) is

$$DR(t, v) = \left\{ \begin{array}{ll} \textit{true} & \textbf{if } (t.Gender = v.Gender) \\ & \textbf{and } (|t.Age - v.Age| < 5) \\ \textit{false} & \text{otherwise} \end{array} \right\}.$$

Partitioning. In this step, each party generates a set of d-dimensional hyper-rectangles and partitions its dataset accordingly. To ensure correctness, every record should be included in at least one hyper-rectangle. A natural way of obtaining these hyper-rectangles is to perform an indexing of the record set. In Section 4 we explore several alternatives, based on BSP-trees, kd-trees and R-trees.

Notice that the structure of the index may reveal sensitive information about data distribution. For example, R-trees do not cover empty regions of the data space. Similarly, kd-trees reveal the median values along the split dimension. To protect against such disclosures, the index itself may be perturbed by the privacy mechanism.

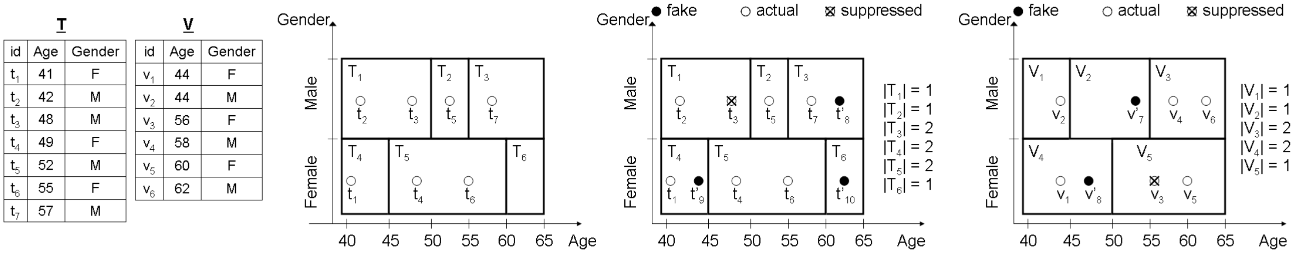
Consider an indexing algorithm that accesses the dataset in terms of range queries only. If the transcript of all queries issued by the algorithm is protected with differential privacy, then the outcome (i.e., the perturbed index) does not leak any private information[4]. This is exemplified in Section 4.2, where we present the *Adaptive-kd* algorithm.

Figure 2(b) shows the set of hyper-rectangles $T_1, \dots, T_6$ generated by party A and the records associated with each hyper-rectangle. The example considers disjoint ranges, such that L_1-sensitivity is easy to compute.

Output Perturbation. To facilitate blocking, each party must disclose to the other party: (1) the extents of the hyper-rectangles obtained from the partitioning step, and (2) the cardinality of each partition. Assuming that the partitioning step respects privacy, the extents can be directly disclosed. However, random noise must be injected into the record counts of every partition (recall that each count is equivalent to answering a $Count$ query over the extent of the corresponding partition). The amount of noise depends on the privacy parameter ϵ as well as the sensitivity of the partitioning step. Specifically, party A determines sensitivity $S_{L_1}^{T}$ and generates

[3] Without loss of generality, throughout the rest of the paper, we will assume that privacy parameters of the parties are equal (i.e., $\epsilon_A = \epsilon_B = \epsilon$).

[4] In effect, this process is equivalent to perturbing the response to a complex query that requests a particular index of the dataset.



T

id	Age	Gender
t_1	41	F
t_2	42	M
t_3	48	M
t_4	49	F
t_5	52	M
t_6	55	F
t_7	57	M

V

id	Age	Gender
v_1	44	F
v_2	44	M
v_3	56	F
v_4	58	M
v_5	60	F
v_6	62	M

(a) Record Sets (b) Hyper-Rectangle Generation and Hashing (c) Output Perturbation (set T) (d) Output Perturbation (set V)

Figure 2: Solution Overview

noise drawn from a Laplace distribution with mean 0 and magnitude $S^T_{L_1}/\epsilon$. Continuing our example, since partitions $T_1, \ldots, T_6$ have disjoint extents, the resulting sensitivity has value 2.[5] Therefore, party A injects Laplace noise with magnitude $2/\epsilon$ into the cardinality of each partition.

Note that, the noise can have both positive and negative values, and is not necessarily integral. Integrality can be addressed by rounding-up to the closest integer value. Positive noise is incorporated by adding fake records to the dataset, while negative noise requires suppressing some of the original records. For instance, in Figure 2(c), record t_3 from rectangle T_1 is removed, whereas t'_8 is added in T_3. Figure 2(c) shows the cardinalities $|T_1|, \ldots, |T_6|$ obtained after perturbation. Figure 2(d) shows the perturbation outcome for record set V.

Adding and suppressing records must not affect matching correctness. When a record is added, it is marked as fake, and assigned remote values (outside attribute domains[6]) so that the decision rule never evaluates to *true* when fake and original records are matched in the SMC phase. Therefore, no information is disclosed to the other party about which partitions contain fake records. We emphasize that fake records are never disclosed other than in encrypted form (i.e., equivalent to releasing only record counts).

On the other hand, every suppressed record is placed in a special partition T' (respectively V'), which is always included in the SMC matching phase against the *entire* dataset of the other party (i.e., the SMC phase is performed for $T' \times V$ and $T \times V'$, respectively). This way, completeness of the matching result is preserved. In the example, $T' = \{t_3\}$ and $V' = \{v_3\}$.

3.2 Blocking

The blocking phase filters out pairs of partitions from the sets $\{T_i\}_{1\leq i\leq m}$ and $\{V_j\}_{1\leq i\leq n}$ that cannot contain matching records based on the given decision rule. Since extents of the partitions are disclosed according to differential privacy, such filtering can be performed efficiently, i.e., without any SMC protocols. For instance, due to the DR condition on attribute $Gender$, none of the records in partitions T_1, T_2 or T_3 can match neither V_4, nor V_5. Similarly, none of T_4, T_5 or T_6 matches V_1, V_2 or V_3.

The decision rule also specifies that the difference between the ages of matching records should be strictly smaller than 5. T_1 has extent $[40, 50)$ on attribute Age, hence it cannot match V_3, which spans Age range $[55, 65)$. Similarly, T_4 with Age range $[40, 45)$ cannot match V_5 which spans Age range $[50, 65)$, etc. After the blocking phase, the following partition pairs are kept as candidates for matching in the SMC phase: (T_1, V_1), (T_1, V_2), (T_2, V_2), (T_2, V_3), (T_3, V_2), (T_3, V_3), (T_4, V_4), (T_5, V_4), (T_5, V_5) and (T_6, V_5). The total number of record pairs that are considered in matching the above partition pairs is 22. Furthermore, the sets of suppressed records T' and V' must also be joined with the entire record set of the other party. Since $|T' \times V| = 6$ and $|T \times V'| = 7$, the total number of records considered in the SMC step is $22 + 13 = 35$. Compared to the cardinality of $|T \times V| = 42$, this represents a saving of $\frac{(42-35)}{42} \times 100 \approx 17\%$.

Considering each partition as a small dataset by itself, any existing solution for privacy preserving record matching can be applied to match the set of non-blocked partition pairs. Section 6 details our SMC solution for matching records of non-blocked partition pairs.

4. PARTITIONING STEP

A partition p consists of a set of points, $Points(p)$ and a d-dimensional hyper-rectangle $Region(p)$ such that for all points t, $t \in Point(p) \Rightarrow t \in Region(p)$. In other words, every point in $Points(p)$ should be contained by the region of partition p. We denote the interval covered by a region r on dimension A_i as $[x_i, y_i]$, where x_i is the lower bound on attribute A_i and y_i is the upper bound. Alternatively, $r[x_i]$ (resp. $r[y_i]$) denotes the lower (resp. upper) bound of region r on A_i.

Given dataset D, a partitioning algorithm outputs a set of partitions $P^D = \{p_1, ..., p_k\}$. We classify partitioning methods into two groups: *space* partitioning and *data* partitioning.

Space partitioning algorithms output partitions that cover the entire data space. Formally,

$$\cup_{i=1}^{k} Region(p_k) = A_1 \times \cdots \times A_d.$$

On the other hand, data partitioning algorithms output partitions that cover the entire dataset:

$$\cup_{i=1}^{k} Points(p_k) = D.$$

Unless records are scattered evenly across the entire data space, partitions generated by data partitioning algorithms tend to leave out empty regions (e.g., T_6 of Figure 2(b)). Such empty regions may leak considerable amounts of sensitive information. For example, following our example in Section 1, if A and B are competitor businesses, B can infer that A's customer base excludes elderly females. To prevent these disclosures, empty regions should be represented as partitions as well and perturbed accordingly.

Another issue that arises with data partitioning methods is the problem of overlapping hyper-rectangles (i.e., partition regions). To ensure correctness, it is sufficient to include a record enclosed by multiple partition regions in only one rather than all.

THEOREM 2. *Let $P^r = \{p_1, \ldots p_k\}$ be the set of partitions whose (overlapping) regions contain the record $r \in T$. Including r in only one partition $p* \in P^r$ does not affect the correctness.*

[5] Here, we assume that the sensitivity of the partitioning step is 0.

[6] Such values can be generated based on domain knowledge (possibly obtained during private schema matching).

PROOF. The proof follows directly from the correctness of the entire protocol. Suppose that for any partition p, all records in $Points(p)$ are correctly classified according to the decision rule after protocol execution. Then, regardless of which partition of P^r includes r, $\forall s \in V, (r, s)$ will be labeled correctly. $\square$

We observe that including r into multiple partitions of P^r will only increase the costs. On the other hand, the overall cost of matching record r with those of V depends on how $p*$ is chosen. In most cases an optimal procedure that specifies the best choice might not be available, leaving parties with heuristic solutions.

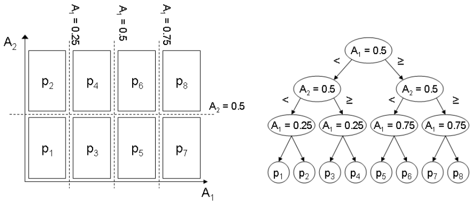
Below, we briefly describe BSP-tree, kd-tree and R*-tree data structures and present our algorithms for releasing a perturbed partitioning of the data.

4.1 BSP-Trees

In d-dimensional space, BSP-trees are constructed by recursively dividing the space into two sub-spaces using $(d-1)$-dimensional hyperplanes. For example in 2D, each step corresponds to heuristically selecting a line and partitioning the space based on this line until certain requirements are met. In 3D, instead of lines, planes are used to split the space. The heuristic method for selecting the splitting hyperplanes depends on the application area, as does the stopping condition.

To simplify the BSP-tree construction process and reduce the data-dependence of the heuristics for splitting hyperplane selection, we only consider axis-orthogonal hyperplanes. The axis is chosen based on the depth of the tree node which is being split. Specifically, if the depth of node n is denoted $Depth(n)$, the splitting hyperplane is orthogonal to axis $Depth(n)\%d + 1$ and splits the space into two equal halves.[7]

Figure 3 exemplifies a BSP-tree of height 3 in 2D space constructed by this heuristic. Data space is first partitioned into equal partitions on A_1 axis through the line $A_1 = 0.5$. For all intermediate nodes at level 1, the next splitting line is orthogonal to A_2, $A_2 = 0.5$. At level 2, the splitting line partitions on A_1 axis again. The resulting partitions are the leaf nodes of the tree: $p_1, ...p_8$. Partitions cover the same volume of space and their regions do not depend on data distribution.



(a) Partition regions

(b) BSP-Tree

Figure 3: BSP-Tree example

Let $\mathcal{Q}^h$ be a query that asks for the BSP-tree index on some dataset D. Since partition extents are built based on attribute domains only, $\mathcal{Q}^h$ is equivalent to a set of 2^h count queries, one for each leaf node's extent. Notice that regardless of h and the distribution of D, the query regions will be disjoint. Therefore, the sensitivity of $\mathcal{Q}^h$ is 2.

4.2 Adaptive construction of kd-trees

kd-trees are another space-partitioning indexing technique. The root node covers the entire space. At each step, a splitting dimension and a split value from the range of the current node on that dimension are chosen heuristically to divide the space into subspaces. Again, the algorithm halts when pre-defined requirements (such as tree height or number of elements covered by a node) are met.

When used as an index for efficient access to spatial objects, a balanced tree where leaf nodes contain approximately the same number of points is of interest. For this purpose, similar to what we have suggested for BSP-tree construction, the splitting dimension dim is chosen based on node depth: $dim = Depth(n)\%d + 1$. The split value on the dimension dim is chosen based on the distribution of the points. Selecting the median value on dim-axis of the points distributes the points evenly among the left and right child nodes.

We start our analysis by calculating the sensitivity of a query $median(A_j)$ that returns the median across attribute A_j.

THEOREM 3. *Let domain of A_j have the range $[A_j^{min}, A_j^{max}]$. Sensitivity of $median(A_j)$ is $(A_j^{max} - A_j^{min})$.*

PROOF. Since the result of a median query is always equal to one of the attribute domain values, the difference between any two median queries (and consequently between two median queries on tables that differ in a single record) is at most $(A_j^{max} - A_j^{min})$. Hence, this is an upper bound for the value of $S_{L_1}^{median(A_j)}$. Next, we show that this is also a lower bound. Consider dataset T with $2q + 1$ records such that the value of the i^{th} record t_i on attribute A_j, $t_i[A]$ is A_j^{min} if $1 \leq i \leq (q + 1)$, and A_j^{max} otherwise. The result of $median(A_j)$ over T is A_j^{min}. Suppose we change the value of the $(q + 1)^{st}$ record from A_j^{min} to A_j^{max}, yielding table T' such that $t_i[A_j] = A_j^{min}$ if $1 \leq i \leq q$, and A_j^{max} otherwise. It follows immediately that the median query for T' returns A_j^{max}. Since T and T' differ in only one record and $|T| = |T'|$, the lower bound on $S_{L_1}^{median(A_j)}$ is $(A_j^{max} - A_j^{min})$. $\square$

The fact that the sensitivity of a median query depends on the domain size may be undesirable for various reasons. Firstly, domain size can be too large for certain attributes (e.g., income). Secondly, any noisy median outside the interval $[A_j^{min}, A_j^{max}]$ is useless for partitioning the data.

One workaround consists of replacing the median query with $(A_j^{min} - A_j^{max})$ count queries, one for each value within the range, and calculating the median based on these noisy counts. The sensitivity of this approach is fixed, $S_{L_1} = 2$, because query regions are disjoint.

If the range of dim is too wide, the noisy median computed by explicit count may be far from the actual one due to noise accumulation. In such cases, an alternative is to use the noisy mean as the split value instead of the median. For a region r, over dimension dim, the following queries are issued:

$$\begin{array}{l} \texttt{Q3}: \texttt{SELECT SUM(t.dim) FROM T AS t WHERE } t \in r; \\ \texttt{Q4}: \texttt{SELECT COUNT}(*) \texttt{ FROM T AS t WHERE } t \in r; \end{array}$$

Here, query $Q3$ retrieves the sum over dim and $Q4$ retrieves the count. Although the responses to both queries will be noisy, the mean can be approximated as $Q3^T/Q4^T$. In our analysis, we assume that the split value is the noisy median (calculated from noisy counts) for categorical attributes and the noisy mean for numerical attributes.

Algorithm 1 provides the entire protocol for adaptive kd-tree construction. The root node covers the entire d-dimensional space. While tree height is below the maximum value h, the next available node is chosen for splitting. The splitting dimension dim is determined by the node depth. For numerical attributes, the split value

[7] The +1 only maps dimension indices from $[0, d-1]$ to $[1, d]$.

Algorithm 1 Adaptive kd-tree construction

```
Require: $h$, maximum height of the tree
  $Region(root) \leftarrow A_1 \times \cdots \times A_d$
  Stack $s \leftarrow \emptyset$
  $s$.Push($root$)
  while Stack is not empty and tree height $\leq h$ do
    Node $n \leftarrow s$.Pop()
    Dimension $dim \leftarrow$Depth($n$)$\%d + 1$
    if $dim$ is numerical then
      $Sum \leftarrow$ Query Sum($dim$) over $Region(n)$
      $Count \leftarrow$ Query Count($*$) over $Region(n)$
      $val \leftarrow Sum/Count$
    else
      for all Categories $c$ in $Region(n)_{dim}$ do
        $Counts[c] \leftarrow$ Query Count($*$) over
          $Region(n) \cap A_1 \times \cdots \times A_{dim-1} \times \{c\} \times \cdots \times A_d$
      end for
      $val \leftarrow$ median calculated from $Counts$
    end if
    $Left(n) \leftarrow$ Left-split of $Region(n)$ on $dim$ at $val$
    $Right(n) \leftarrow$ Right-split of $Region(n)$ on $dim$ at $val$
    $s$.Push($Left(n)$), $s$.Push($Right(n)$)
    Adjust tree height
  end while
```

val is the noisy mean. For categorical attributes, val is the noisy median.

If the sensitivity of the set of queries exceeds the preset threshold $\epsilon\lambda$ during execution of Algorithm 1, according to differential privacy, the database should not respond with an answer. To prevent this from happening, we now calculate an upper bound on the sensitivity of Algorithm 1. The noise added to each partition will be computed according to this bound.

THEOREM 4. *Let $\mathcal{Q}^h = \{Q_1^h, ..., Q_q^h\}$ be the query set that adaptively constructs a kd-tree of height h. The largest subset $\mathcal{Q}_*^h \subseteq \mathcal{Q}^h$ of queries with overlapping query regions has size*

$$\begin{aligned} n_h &= 2 \times \sum_{i=0}^{h-1} I(A_{i\%d+1} \textit{ is numerical}) \\ &+ \sum_{i=0}^{h-1} I(A_{i\%d+1} \textit{ is categorical}). \end{aligned}$$

PROOF. When $h = 1$, the kd-tree contains 2 leaf nodes and the root node. $\mathcal{Q}^1$ consists of the queries that determine the split value of the root on attribute A_1. If A_1 is categorical, then $\mathcal{Q}^1$ is a set of non-overlapping count queries and $n_1 = 1$. On the other hand, if A_1 is numerical, $\mathcal{Q}^1$ consists of a sum and a count query over the entire data space and $n_1 = 2$. Using indicator functions $I(.)$, n_1 can be expressed as

$$n_1 = 2 \times I(A_1 \text{ is numerical}) + I(A_1 \text{ is categorical}).$$

Now consider the case $h = i > 1$. Assume that $|\mathcal{Q}_*^{h-1}| = n_{h-1}$ holds. We want to show that $|\mathcal{Q}_*^h| = n_h$.

Adding another level to the tree requires splitting the current leaf nodes. Based on our dimension selection heuristic, every node at level $h-1$ has to be split on the same attribute $A_{(h-1)\%d+1} = A^{\#}$.

Suppose $A_{\#}$ is categorical. The queries that determine the split values are exactly the set $\mathcal{Q}^h - \mathcal{Q}^{h-1}$. However, none of these queries overlap with each other because by definition of space partitioning algorithms, any two nodes at the same level cover disjoint areas of the data space. Therefore, at most one new query is added to $\mathcal{Q}_*^{h-1}$. In other words, when $A_{\#}$ is categorical, $n_h = n_{h-1} + 1$.

If $A_{\#}$ is numerical, B asks two overlapping queries (sum and count) per leaf node to determine the split values. Again, queries regarding sibling nodes do not overlap with each other. In this scenario, $\mathcal{Q}_*^{h-1}$ grows by 2.

Accordingly, we can calculate n_h as

$$\begin{aligned} n_h &= n_{h-1} + 2 \times I(A_1 \text{ is num.}) + I(A_1 \text{ is cat.}) \\ &= 2 \times \sum_{i=0}^{h-1} I(A_{i\%d+1} \text{ is num.}) + \sum_{i=0}^{h-2} I(A_{i\%d+1} \text{ is cat.}). \end{aligned}$$

□

Next, we formulate the sensitivity of $\mathcal{Q}^h$.

THEOREM 5. *L_1-sensitivity of $\mathcal{Q}^h$ is $2n_h$.*

PROOF. The proof follows directly from the definition of L_1-sensitivity. Changing one record of the dataset changes the result of count queries by at most 2. For sum queries, assuming the numerical attribute in question is normalized to $[0, 1]$, the maximum distance in L_1-norm corresponds to changing some record's attribute value from 0 to 1. Here again, the L_1-norm distance changes by at most 2. Normalization can be performed as a preprocessing step[8]. In total, adaptively building the kd-tree (or equivalently retrieving the partition regions) has sensitivity $2n_h$. □

In order to obtain the partition cardinalities themselves, query counts over all partition regions should be issued too. Including these queries, the overall sensitivity of the partitioning step using Algorithm 1 becomes $2n_h + 2$. $\mathcal{Q}_*^h$ grows by only one query because partitions do not overlap with each other but only with their parents. Therefore changing one record affects at most 2 additional query responses. Hence, the overall sensitivity is $2n_h + 2$.

4.3 Data Partitioning Index Structures

So far, we have considered only space partitioning indexing techniques. However, many data partitioning indexing methods exist (e.g., R-trees, TV-trees, X-trees, M-trees, etc) that usually obtain better record clustering accuracy than their space partitioning counterparts. In our work, we consider R^*-trees [5], an R-tree variation with good clustering properties. Note that, in general, there are no guarantees that the leaf set will completely cover the data space. To satisfy differential privacy, an additional partition p' is added to the transcript, which covers the entire data space and includes all data points. A 2-bound of L_1-sensitivity is computed with respect to all leaf nodes and p', as described in Section 2.1.

Note that, in most cases, data partitioning index structures do not fulfill the condition that distinct leaf nodes have disjoint extents in the attribute domain space. Due to overlaps, the L_1-sensitivity of the resulting leaf set may have large values (recall from Section 2.1 that sensitivity is proportional to the amount of overlap in query ranges). Therefore, a trade-off emerges: on one hand, the leaf nodes have smaller extents, leading to better matching accuracy. On the other hand, the sensitivity is increased, requiring more noise to be added to the reported cardinalities, hence decreasing blocking efficiency. Our experimental results (Section 7) show that the latter factor has a very high negative influence, indicating that sets of overlapping hyper-rectangles are not suitable for private matching in the differential privacy context.

[8] We assume that the ranges of continuous attributes are determined before the record linkage process (i.e., during schema matching).

5. BLOCKING STEP

Given two regions R_1 and R_2, let $d_i^{inf}(R_1, R_2)$ denote the infimum distance between any pair of records within R_1 and R_2 over the i^{th} dimension. Formally,

$$d_i^{inf}(R_1, R_2) = inf_{\substack{t \in R_1 \\ v \in R_2}}(dist_i(t, v)).$$

By definition, $d_i^{inf}(R_1, R_2)$ is the greatest lower bound on the distance. If $d_i^{inf}(R_1, R_2) > \theta_i$ for some $1 \leq i \leq d$, then any two points from R_1 and R_2 cannot match.

We define the supremum distance similarly:

$$d_i^{sup}(R_1, R_2) = sup_{\substack{t \in R_1 \\ v \in R_2}}(dist_i(t, v)).$$

Supremum distance function helps matching two regions. By definition, $d_i^{sup}(R_1, R_2)$ limits from above the maximum distance between two arbitrary points of R_1 and R_2. If these distance values never exceed the threshold for any attribute, then all points within $R_1 \times R_2$ should match.

Based on infimum and supremum distance functions, the blocking decision rule $BDR(R_1, R_2)$ can be defined as follows:

$$BDR(R_1, R_2) = \left\{ \begin{array}{cc} \text{N if } \exists\, 1 \leq i \leq d \; d_i^{inf}(R_1, R_2) > \theta_i & \\ \text{M if } \forall\, 1 \leq i \leq d \; d_i^{sup}(R_1, R_2) \leq \theta_i & \\ \text{U} & \text{otherwise} \end{array} \right\}$$

Here the return values M, N and U refer to *match, non-match* and *undecided* respectively. As exemplified in Section 3.2, not all pairs of regions can be classified as M and N. Whenever an accurate decision cannot be drawn, the pair is labeled U. Records belonging in such regions will be labeled privately through the SMC protocols described in Section 6.

5.1 Overall protocol for blocking

Let $\{T_i\}_{1 \leq i \leq m}$ (resp. $\{V_j\}_{1 \leq j \leq n}$) be the set of partitions extracted from party A's (resp. B's) data. Let us denote the region covered by a partition p as $Region(p)$ and the perturbed set of records within the region as $Points(p)$. Recall from Section 4 that all suppressed records are returned in special additional partitions T' of T and V' of V.

Algorithm 2 Protocol for the blocking step

```
Require: $T = \{T_i\}_{1 \leq i \leq m} \cup T'$ and $V = \{V_j\}_{1 \leq j \leq n} \cup V'$
 1: for all Partitions $T_i \in T$ do
 2:     for all Partitions $V_j \in V$ do
 3:         if $BDR(Region(T_i), Region(V_j)) = U$ then
 4:             Privately match $Points(T_i) \times Points(V_j)$
 5:         else if $BDR(Region(T_i), Region(V_j)) = M$ then
 6:             Add $Points(T_i) \times Points(V_j)$ to the result
 7:         end if
 8:     end for
 9: end for
10: Privately match $Points(T') \times V$
11: Privately match $Points(V') \times T$
```

Algorithm 2 describes the overall protocol for the blocking step. For every partition $\{T_i\}_{1 \leq i \leq m}$ of T and $\{V_j\}_{1 \leq j \leq n}$ of V, the blocking decision rule BDR is evaluated. In step 3, record pairs that will be labeled with SMC protocols are identified. Step 5 inserts matching record pairs to the result set. Finally, in steps 10 and 11, suppressed records are labeled.

6. SMC STEP

For pairs of records that are not blocked, it is necessary to securely determine whether $dist_i(t[i], v[i]) \leq \theta_i$, for all attributes A_i. The decision rule can be evaluated securely using generic SMC circuit evaluation techniques [14]. However, less expensive techniques can be employed, such as commutative encryption [2] and homomorphic encryption [18]. In our work, we use a protocol that relies on the latter[9].

Let $E_{pk}(.)$ denote the encryption function with public key pk and $D_{pr}(.)$ the decryption function with private key pr. A secure public key cryptosystem is called homomorphic if it satisfies the following requirements: (1) Given ciphertexts $E_{pk}(m_1)$ and $E_{pk}(m_2)$ of messages m_1 and m_2, there exists an efficient algorithm to compute the public key encryption of $m_1 + m_2$, denoted as $E_{pk}(m_1 + m_2) := E_{pk}(m_1) +_h E_{pk}(m_2)$. (2) Given a constant k and the encryption of m_1, $E_{pk}(m_1)$, there exists an efficient algorithm to compute the public key encryption of km_1, denoted as $E_{pk}(km_1) := k \times_h E_{pk}(m_1)$.

Using homomorphic encryption, L_1 distances can be immediately determined. We focus further on how to privately compute Euclidean distances, i.e., $dist_i(t[i], v[i]) = (t[i] - v[i])^2 = (t[i])^2 - 2 \times v[i] \times t[i] + (v[i])^2$. Party A creates a homomorphic public/private key pair, and sends the public key to party B. A also sends to B encrypted values $\{t_{id}, E_{pk}((t[i])^2), E_{pk}(-2 \times t[i])\}$ where t_{id} is a randomly generated record identifier for record t. Next, party B computes for each of its records v_{id} the value $E_{pk}((t[i])^2) +_h (E_{pk}(-2 \times t[i]) \times_h v[i]) +_h E_{pk}((v[i])^2)$ which is equal to $E_{pk}((t[i] - v[i])^2)$. Party B creates message $\{r_{id}, E_{pk}((t[i] - v[i])^2)\}$ for each record pair comparison, and keeps a local entry stating that the comparison with random identifier r_{id} is comparing records identified by t_{id} and v_{id}. To prevent party A from learning how many records of B are candidate matches for each hyper-rectangle disclosed by A, party B shuffles randomly the messages and sends them to A. Note that, such secure distance evaluation could be combined with a secure comparison protocol, such that the actual distance is not revealed, but only its relative value to threshold θ_i. Using the private key, party A can decrypt the distance value, and find out the r_{id} of the matching records. The selected r_{id} values are sent to party B, which outputs the matching identifier pairs t_{id} and v_{id}.

6.1 Limited SMC budget

In the record linkage community, efficiency of a blocking scheme is measured by the *reduction ratio* metric [10]. Given a baseline comparison space $\mathcal{S}$, reduction ratio (RR) is the fraction of savings from the comparison space attained by the blocking scheme. We compare our results to a naive solution that privately evaluates DR over all record pairs in the Cartesian product $T \times V$.[10] Therefore, $|\mathcal{S}| = |T \times V| = |T| \times |V|$. Then, RR can be defined formally as

$$RR = 1 - \frac{\text{number of decision rule evaluations}}{|T| \times |V|}.$$

When the input datasets T and V are large, even after considerable amount of reduction in comparison space, costs of applying our solutions might still be higher than the amount anticipated by the participants. In order to prevent such concerns related to high costs from hampering the record matching process, we now discuss an extension to our methods where participants can determine an upper bound on the number of SMC protocol invocations

[9] Please note that our framework could be used with any SMC protocol.

[10] Existing SMC protocols usually compare all record pairs.

that evaluate the decision rule DR. We call this upper bound the SMC_budget.

Similar to RR, we represent SMC_budget as a fraction of the Cartesian product size. For example, if $|T| = |V| = 10^3$, then $|T \times V| = 10^6$ and $SMC\_budget = 0.01$ implies DR will be evaluated at most $10^6 \times 0.01 = 10^4$ times using the SMC protocols of Section 6.

Notice that all record pairs that were unlabeled after the blocking step are labeled in the SMC step. In this case, SMC protocols need to compare $(1-RR)\times|T|\times|V|$ pairs. If $1-RR \leq SMC\_budget$, then there is no challenge in enforcing the limits over SMC operations because the budget meets or exceeds the need. However, when $1 - RR > SMC\_budget$, then some record pairs cannot be properly labeled.

In order to prevent disclosure of irrelevant record pairs, we assume that all such records are excluded from the result set (i.e., assumed to be mismatching record pairs). Whenever SMC_budget is insufficient, record pairs should be chosen carefully to maximize the number of matching record pairs found in the SMC step. This notion is captured by the *recall* measure. Let H be some heuristic that guides us in selecting the record pairs towards which the SMC_budget is spent. Then the recall of H, denoted $Recall_H$, is the fraction of matching record pairs that H can identify in the SMC step. Formally, denoting matching record pairs by the set n_M, the recall of heuristic H is defined as

$$Recall_H = \frac{\text{number of matching pairs found by } H}{|n_M|}.$$

A naive approach to enforce SMC_budget would be choosing a random subset of unlabeled record pairs. Yet, it makes more sense to use the information contained in the partition regions. Below we discuss various heuristics that help identify possibly matching record pairs. Among these, the heuristic that has maximum recall should be favored.

6.2 Selection heuristics

We propose the following heuristics. An empirical evaluation of these heuristics is provided in Section 7.6.

H_1**- Minimum comparison cost first.** In this heuristic, partitions of T are sorted with respect to the number of secure DR evaluations required to find all matching records of V. Then, the partitions are processed in ascending order. The idea is maximizing the fraction of records of T that are matched against V. H_1 would be advantageous if the partitions were weighted based on some criteria. For example, in a hospital dataset, matching cancer patients might be given priority over patients of less lethal illnesses.

H_2**- Minimum volume partition first.** In this heuristic, partitions p of T are sorted with respect to the volume of their regions, $Region(p)$. Then, partitions are processed in ascending order. Considering records as random variables supported over their partition regions, this heuristic assumes that lower volumes imply less uncertainty in estimating the actual value of a record. Based on this idea, partitions with the smallest region are processed first.

H_3**- Partition pair** (p_1, p_2) **with maximum** $Region(p_1)$ $\cap Region(p_2)$ **volume first.** This heuristic assumes that the volume of the intersection between partition regions is an accurate indicator of possibly matching partition pairs. Therefore pairs of partitions are ordered based on the intersection volume and processed in descending order.

7. EXPERIMENTAL ANALYSIS

Our solutions aim at reducing the costs of private record matching. Therefore a natural metric of efficiency is the reduction in the number of SMC operations compared to the baseline solution that privately evaluates DR over all record pairs. This notion is captured by the reduction ratio, RR, defined in Section 6.1.

Apart from reduced costs, another important advantage of our solutions (or hybrid methods, in general) is the efficient retrieval of possibly matching record pairs under limited SMC resources. In Section 6.2, we suggested several selection heuristics to perform this task. Our secondary measure is the recall of such heuristics, defined in Section 6.1.

Experimental results involving differential privacy exhibit high variance due to the randomness involved. Therefore we provide results averaged over 10 runs, each initialized with distinct seeds of randomness.

We used the real-world $Census - income$ dataset from the UCI Machine Learning Repository[11]. This dataset has been used heavily in privacy preserving data analysis research and also facilitates easy comparison of our results against [16]. As preprocessing, we first combined the training and test datasets. Then all records with missing attribute values were removed. In order to build two datasets, we shuffled the remaining 142,521 records and partitioned them into 3 subsets: d_1, d_2, d_3 containing 47,507 records each. Then we merged d_1 and d_2 to build the first dataset D_1 and, d_2 and d_3 to build the second dataset D_2. This way regardless of the matching thresholds of the decision rule, the records in $D_1 \cap D_2 = (d_1 \cup d_2) \cap (d_2 \cup d_3) = d_2$ match one another.

The $Census - income$ dataset contains 40 different attributes. Among these, we have selected the following quasi-identifying attributes: numerical *age* and categorical *education, marital status, race* and *sex* attributes. The reason why we select quasi-identifying attributes should be clear: by nature, quasi-identifiers help identify similar records in absence and/or irrelevance of unique identifiers.

For categorical attributes, we set the matching threshold θ to 0 and use Hamming distance. For numerical attributes, after normalization to $[0, 1]$, the matching threshold is set to 0.05 and the distance is measured by Euclidean distance.

Another important experiment parameter is the ϵ of ϵ-differential privacy. Given the L_1 sensitivity $S_{L_1}(Q)$ of a query set Q, ϵ-differential privacy requires that Laplace noise with mean 0 and magnitude λ be added to the query results such that $S_{L_1}(Q) \leq \epsilon\lambda$. With fixed sensitivity, smaller values of ϵ imply adding more noise and consequently higher levels of privacy. The default value of ϵ in our experiments is 0.3.

We implemented all partitioning algorithms discussed in Section 4. For R*-tree experiments, results with 3 different page sizes (2K, 4K and 8K bytes each) are presented. Page sizes are indicated next to the labels (e.g., R*-tree (2K) represents the results for pages of size 2K bytes). The default heights of BSP-trees and Adaptive-kd trees are set to 10.

In addition to the partitioning algorithms, we also provide some results obtained with the hybrid private record linkage method described in [16]. To this end, we implemented the Mondrian k-anonymization method of [19]. The results for these experiments are labeled as *Mondrian* in the figures. The default value of k in our experiments is 250.

7.1 Dataset Size

In this experimental scenario, we incremented the sizes of the input datasets T and V that are linked. Fractions on the x-axis of Figure 4 indicate the proportion of the preprocessed dataset from which T and V are built. Thus, $1/1$ implies the entire *Census-income* dataset was used in the experiment.

[11] http://www.ics.uci.edu/ mlearn/MLRepository.htm

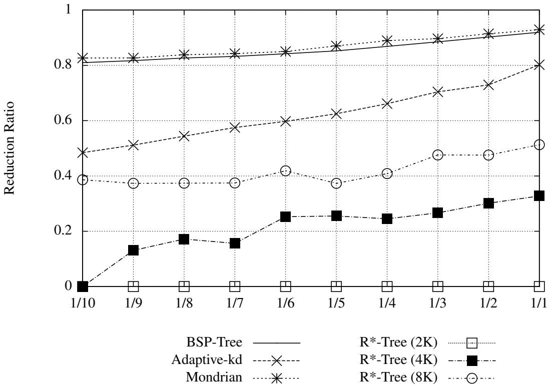


Figure 4: Reduction ratio vs. dataset size

As expected, increasing the dataset size without changing the privacy parameter ϵ increases the reduction ratio. The height of BSP-trees and Adaptive-kd trees is fixed in this experiment scenario. At fixed height, the number of partitions obtained is the same irrespective of the dataset size. Since the regions covered by the partitions remain approximately (exactly for BSP-trees) the same as well, adding new records simply increases the number of records in each partition. With more records in every partition and fixed ϵ, disadvantages associated with the added noise decrease. Hence, the reduction ratio improves.

Note that, the R^*-tree partitioning performs quite poorly, due to the overlap between leaf nodes, which increases L_1-sensitivity. The effect of the overlap is more pronounced for small block sizes ($2K$), as the number of leaf nodes is larger. In addition, a small node capacity means that the effect of the added noise is more significant, compared to the block size. As block size increases, the performance of the R^*-tree partitioning improves, but it is still considerably worse than its non-overlapping indexing counterparts. We do not consider R^*-trees further.

7.2 Number of Attributes

Figure 5 reports the reduction ratio as a function of increasing number of attributes. In the experiments, an attribute set of size $n : 1 < n < 10$ consists of the first n elements of the set *age, education, marital status, race, sex, wage per hour, capital gains, capital losses, dividends from stocks*.

According to our results, reduction ratio does not vary much with the number of attributes. While the values for BSP-tree and Adaptive-kd tree experiments fluctuate considerably, this is evident from the results with Mondrian. For 6, 7 and 8 attributes, RR of Mondrian remains the same. The variance in BSP and Adaptive-kd tree measurements stem from the randomness introduced by the privacy mechanism.

7.3 Privacy Level: ϵ

The parameter ϵ is inversely related to the privacy guarantees of differential privacy. In other words, at lower values of ϵ, Laplace noise of larger magnitude is added to query responses. Consequently one would expect less efficient blocking as ϵ declines. This finding is also supported by Figure 6.

7.4 Tree height, h

Our results with varying height of BSP and Adaptive-kd trees are provided in Figure 7. Interestingly, the reduction ratio initially improves as the trees grow deeper. But above some threshold value, building even deeper trees damages the efficiency of blocking.

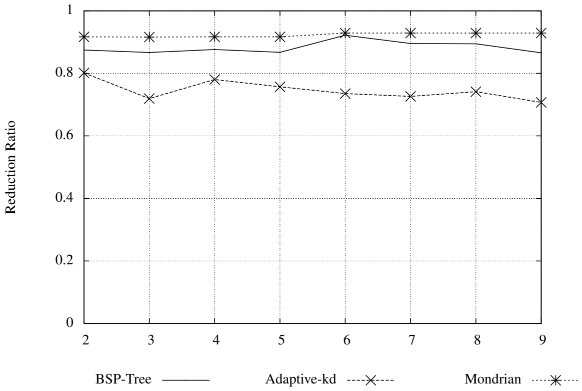


Figure 5: Reduction ratio vs. number of attributes

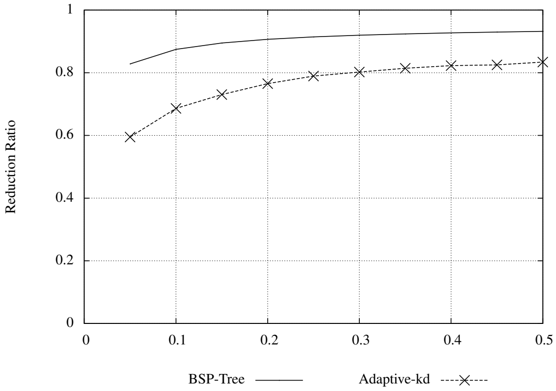


Figure 6: Reduction ratio vs. privacy level, ϵ

The number of partitions generated by our space partitioning trees is bounded from above by 2^h. For small values of h, only a few partitions are output in the partitioning step. In order to cover the entire space, each of these partitions typically covers large volumes of space and contains large amounts of records. Efficiency of our blocking step depends on the granularity of partition regions. Higher granularity partitions of deeper trees perform better in the blocking step. That's why up to a certain threshold ($h = 9$ for BSP-trees and $h = 6$ for Adaptive-kd trees), reduction ratio improves.

The height h that maximizes reduction ratio depends on two parameters: sensitivity of the partitioning algorithm and granularity of the partition regions. As h increases, partitions cover smaller volumes and contain less records. However, sensitivity either does not change (i.e., BSP-trees) or increases (i.e., Adaptive-kd trees). In both cases, while partition sizes decline, magnitude of the added noise does not. Therefore the ratio of fake records (added to the partitions) and suppressed records (removed from the partitions) to actual records increases, and reduction ratio deteriorates. Since sensitivity of the BSP-tree algorithm is less than Adaptive-kd tree algorithm, this occurs at a larger value of h.

Notice that the reduction ratio of BSP-trees does not change after $h = 16$. This is because for $h > 16$, the tree height saturates, i.e., there is no allowable split. The same situation occurs with the Adaptive-kd experiments as well, but in this case sensitivity of the queries are pre-set based on h. Therefore h still affects the results. Since larger h implies higher sensitivity, the same partitions are perturbed with larger and larger magnitudes of noise. Consequently, the reduction ratio of Adaptive-kd keeps declining as h increases.

7.5 Minimum partition size

In this experiment scenario we compare our hybrid solutions that

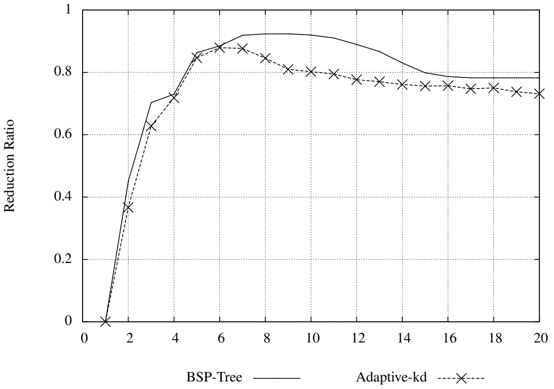


Figure 7: Reduction ratio vs. maximum tree height

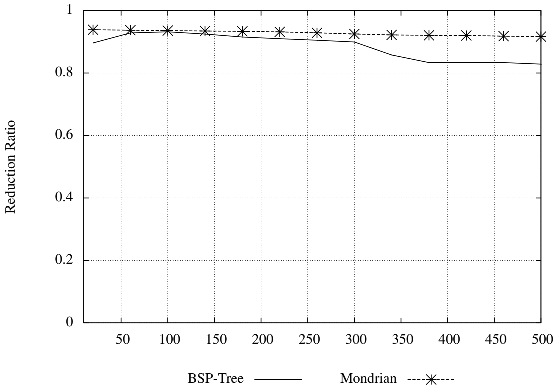


Figure 8: Reduction ratio vs. minimum partition size

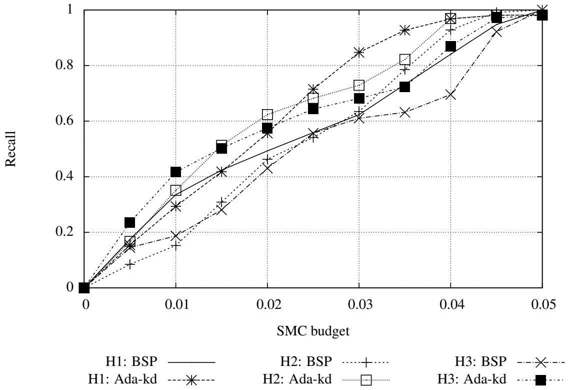


Figure 9: Recall of various selection heuristics

rely on differential privacy to the hybrid solutions of [16] based on anonymization. For this purpose, we use BSP-trees as the partitioning algorithm and Mondrian as the anonymization method. Figure 8 provides the results. In each experiment, the x-axis value corresponds to the parameter k of k-anonymity for Mondrian.

In order to enforce the minimum size requirement on BSP-trees, we set the maximum height h to infinity and let the tree grow indefinitely. At the lowest level of intermediate nodes that violates the minimum partition size requirement, tree growth is halted and partitions are generated from the nodes at the previous level (say, l). Since magnitude of the Laplace noise (of differential privacy) does not depend on h, this method outputs exactly the same tree as the method described in Section 4.1 would with height parameter $h = l$.

Unfortunately, a similar approach fails with Adaptive-kd trees. As described in Section 4.2, sensitivity of the adaptive algorithm depends heavily on the tree height h. We do not know apriori, which value of h will suffice to obtain the maximum-height tree that does not violate the partition size condition. Obviously setting h to a considerably high value would solve the problem, but then again, the comparison would not be fair. That's why we do not provide any results with the Adaptive-kd algorithm.

According to Figure 8, the relationship between reduction ratio and minimum partition size is very similar to the relationship between reduction ratio and tree height h (see Figure 7). This result is rather intuitive. Satisfying a larger partition size requirement is only possible if the tree growth is halted at an earlier stage.

The reduction ratio obtained by Mondrian is always higher than that of the BSP-trees. Mondrian is also more resistant against larger values of the minimum size requirement. This is an inevitable result of the stronger privacy guarantees of differential privacy. Even if the two methods generate the same partitioning of the data space, our solution using BSP-trees suffers from the added noise.

7.6 Selection Heuristics

We implemented the selection heuristics proposed in Section 6.2. In Figure 9 we provide the results with varying SMC_budget. With both methods, increasing the SMC budget allows more private evaluations of the decision rule. As expected, for higher values of SMC_budget, recall improves. Every heuristic provides perfect recall when $SMC\_budget = 1 - RR$ for the corresponding method. At $SMC\_budget = 0$, SMC operations are not allowed, hence matching can only be done in the blocking step. Unfortunately, none of the algorithms finds any matches and recall is 0.

At any value of $SMC\_budget > 0$, recall depends heavily on the reduction ratio. When RR is high, the same SMC budget covers a larger fraction of the un-blocked records pairs. Therefore, one would expect BSP-trees to achieve higher recall than Adaptive-kd. However, this is not the case. In fact, the recall of Adaptive-kd is almost always higher than BSP-trees (except lower SMC budgets with H_1). The reason is the way Adaptive-kd selects the split values. By choosing the median as the split value, Adaptive-kd takes into account the density of records. BSP, on the other hand, makes the decision based purely on the domain. Therefore, Adaptive-kd is able to create smaller partitions in dense areas and yield higher recall.

Among the three heuristics, the highest recall by BSP-trees is achieved through H_1 (i.e., min. comparison cost first). Since all partition regions are of the same volume, H_2 (i.e., min. volume partition first) cannot distinguish one partition pair from another. H_3 (i.e., max. intersection volume first) has similar problems: two partitions either entirely overlap or do not intersect at all.

With Adaptive-kd-trees, the generated partitions have various volumes and can be located arbitrarily within the data space (recall that BSP outputs a tiling of the space). In this setting, H_2 performs more consistently and on average better than both H_1 and H_3. For smaller SMC budgets, H_3; and for bigger budgets H_1 performs better.

8. RELATED WORK

Record linkage has been studied for more than four decades [11]. However, few methods for *private* record matching have been investigated. Some initial approaches are motivated by the privacy requirements of e-health applications [6]. The work most closely related to ours is by Al Lawati et al. [4] who propose a secure blocking scheme to reduce costs. However, their approach works only for a specific comparison function, whereas ours can be used with several such functions.

In addition to the above approaches, there are two major areas that are closely related to our work, even though no work in such areas addresses our exact problem: secure set intersection and private data sharing. Several approaches have investigated the secure set intersection problem [18]. Secure set intersection methods deal with exact matching and are too expensive to be applied to large databases due their reliance on cryptography. Furthermore, these protocols deal with the intersection of sets of simple elements, and are not designed for exploiting the semantics behind database records. Agrawal et al. [2] formalize a general notion of private information sharing across databases that relies on commutative encryption techniques. This work has opened the way to many other related protocols [12, 1].

Cryptographic methods usually require extensive communication and computation between participants. An alternative that has recently become popular is anonymization. Anonymization techniques rely on the fact that privacy of sensitive data is a concern only if the individuals related to the data can be identified. However, removing personal identifiers does not always protect individuals against disclosure of identity. The most popular solution to the anonymity problem is k-anonymity, which requires that an individual should be indistinguishable from at least $(k-1)$ others [23].

Inan et al. present a hybrid method for private record linkage that combines cryptographic methods and anonymization methods [16]. Their approach is based on k-anonymizing the input datasets and linking as many record pairs as possible through the k-anonymous versions released by the data holders. At reasonable levels of anonymity, determined by the privacy parameter k, this hybrid approach attains over 90% savings in terms of costly cryptographic operations. However, the privacy guarantees are limited to those of k-anonymity. In this study, we propose another hybrid method that obtains comparable savings under a much stronger definition of privacy, namely differential privacy.

9. CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel approach that combines differential privacy and cryptographic methods to solve the private record linkage problem. Our method allows participants to trade-off between accuracy, privacy and costs. As future work, we will experiment with other multidimensional indexes (such as R+-trees) and consider their alternatives (e.g., clustering).

10. ACKNOWLEDGMENTS

This work was partially supported by National Science Foundation Grants Career-0845803 and CNS-0716424 and, Air Force Office of Scientific Research MURI Grants FA9550-08-1-0265 and FA9550-07-1-0041.

11. REFERENCES

[1] R. Agrawal, D. Asonov, M. Kantarcioglu, and Y. Li. Sovereign joins. In *ICDE '06*, page 26, 2006.

[2] R. Agrawal, A. Evfimievski, and R.Srikant. Information sharing across private databases. In *Proceedings of ACM SIGMOD 2003*.

[3] R. Agrawal and R. Srikant. Privacy preserving data mining. *Proceedings of ACM SIGMOD*, pages 439–450, 2000.

[4] A. Al-Lawati, D. Lee, and P. McDaniel. Blocking-aware private record linkage. In *Proceedings of IQIS 2005*, Baltimore, Maryland, 2005.

[5] N. Beckmann, H.-P. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. *SIGMOD Rec.*, 19(2):322–331, 1990.

[6] T. Churces and P. Christen. Some methods for blindfolded record linkage. *Medical Informatics and Decision Making*, 4(9), 2004.

[7] C. Clifton, M. Kantarcıoğlu, X. Lin, J. Vaidya, and M. Zhu. Tools for privacy preserving distributed data mining. *SIGKDD Explorations*, 4(2):28–34, Jan. 2003.

[8] C. Dwork. Differential privacy. In *ICALP (2)*, pages 1–12, 2006.

[9] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[10] M. G. Elfeky, A. K. Elmagarmid, and V. S. Verykios. TAILOR: A record linkage tool box. In *ICDE '02*, pages 17–28, 2002.

[11] A. Elmagarmid, G. Panagiotis, and S. Verykios. Duplicate record detection: A survey. *IEEE TKDE*, 19(1), 2007.

[12] F. Emekci, D. Agrawal, A. El Abbadi, and A. Gulbeden. Privacy preserving query processing using third parties. In *Proceedings of ICDE 2006*, Atlanta, GA, 2006.

[13] M. J. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Eurocrypt 2004*, May 2-6 2004.

[14] O. Goldreich. *The Foundations of Cryptography*, volume 2, chapter General Cryptographic Protocols. Cambridge University Press, 2004.

[15] B. A. Huberman, M. Franklin, and T. Hogg. Enhancing privacy and trust in electronic communities. In *Proceedings of the First ACM Conference on Electronic Commerce (EC99)*, pages 78–86, 1999.

[16] A. Inan, M. Kantarcioglu, E. Bertino, and M. Scannapieco. A hybrid approach to private record linkage. In *ICDE '08*, pages 496–505, 2008.

[17] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin. A cryptographic approach to securely share and query genomic sequences. *IEEE Transactions on information technology in biomedicine*, To appear.

[18] L. Kissner and D. Song. Privacy-preserving set operations. In *Advances in Cryptology — CRYPTO 2005*, 2005.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE '06*, pages 25–35, 2006.

[20] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE 2007*, pages 106–115.

[21] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ICDE 2006*, page 24, 2006.

[22] M. Scannapieco, I. Figotin, E. Bertino, and A. Elmagarmid. Privacy preserving schema and data matching. In *SIGMOD*, page 653, 2007.

[23] L. Sweeney. k-anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-Based Systems*, 10:557–570, 2002.

[24] J. Vaidya and C. Clifton. Secure set intersection cardinality with application to association rule mining. *Journal of Computer Security*, 13(4), Nov. 2005.

[25] X. Xiao and Y. Tao. Output perturbation with query relaxation. *PVLDB*, 1(1):857–869, 2008.