Privacy-Preserving Matching of Spatial Datasets with Protection against Background Knowledge

Gabriel Ghinita[1,2], Carmen Ruiz Vicente[3], Ning Shang[2] and Elisa Bertino[1,2]
[1]Purdue Cyber Center and [2]Dept. of Computer Science, Purdue University
email: {gghinita,nshang,bertino}@cs.purdue.edu
[3]Aalborg University, email: carmrui@cs.aau.dk

ABSTRACT

Private matching (or join) of spatial datasets is crucial for applications where distinct parties wish to share information about nearby geo-tagged data items. To protect each party's data, only joining pairs of points should be revealed, and no additional information about non-matching items should be disclosed. Previous research efforts focused on private matching for relational data, and rely either on space-embedding or on SMC techniques. Space-embedding transforms data points to hide their exact attribute values before matching is performed, whereas SMC protocols simulate complex digital circuits that evaluate the matching condition without revealing anything else other than the matching outcome.

However, existing solutions have at least one of the following drawbacks: *(i)* they fail to protect against adversaries with background knowledge on data distribution, *(ii)* they compromise privacy by returning large amounts of false positives and *(iii)* they rely on complex and expensive SMC protocols. In this paper, we introduce a novel geometric transformation to perform private matching on spatial datasets. Our method is efficient and it is not vulnerable to background knowledge attacks. We consider two distance evaluation metrics in the transformed space, namely L_2 and L_∞, and show how the metric used can control the trade-off between privacy and the amount of returned false positives. We provide an extensive experimental evaluation to validate the precision and efficiency of our approach.

Categories and Subject Descriptors

H.2.0 [**General**]: Security, integrity, and protection; H.2.8 [**Database applications**]: Spatial databases and GIS

General Terms

Security, Performance

Keywords

Location Privacy, Privacy-Preserving Data Linkage

ACM GIS '10 , November 2-5, 2010. San Jose, CA, USA (c) 2010 ACM
ISBN 978-1-4503-0428-3/10/11...$10.00 .

1. INTRODUCTION

Private matching (or join) of spatial datasets has several important practical applications. Consider for instance two parties: a cable television company A and an Internet provider B that operate in the same city. Both companies want to expand their customer base, and are considering merging their operations. However, a merger is financially feasible only if the existing wired-connection infrastructure can be re-used to a certain extent. This decision can be evaluated by asking a spatial join of the form *"find the pairs of customers of parties A and B that are separated by a distance below* 200 *meters"*. The query result reveals to the two parties the sets of users that can be offered a combined television and Internet service. However, in the process of query evaluation, locations of customers that are *not* part of the join result may also be revealed to the other party. Such disclosure is not desirable, since the customer base is an important asset to each company. For instance, party B may act maliciously and sell the information about the points of A to a competitor cable television company. Therefore, it is important to perform the join in a privacy-preserving fashion, in the sense that no additional data points other than the ones included in the join result should be disclosed in the matching process.

Several techniques for private matching have been proposed. These can be categorized into secure multi-party computation (SMC) and data mapping approaches. SMC techniques for private joins [6] belong to a broader family of protocols that can evaluate privately any function $\mathcal{F}$ defined on inputs held by parties A and B. The idea is to represent $\mathcal{F}$ as a binary circuit, and privately evaluate the value of the circuit using randomized shares that are exchanged by the two parties. SMC techniques are very powerful from a theoretical standpoint, and provide strong security. The join results are accurate, and only the two parties A and B are involved in the protocol (i.e., no third party is required). However, the computational and communication cost are extremely high, and, as highlighted in previous work [8], no practical SMC-based private join solutions exist.

Data mapping techniques hide the data of both parties according to geometrical [14] or algebraic [17] transformations. Such techniques conjecture that it is difficult for an adversary to learn the exact coordinates of a point based on the point's mapping. However, the mapping affects the precision of query evaluation, and it is possible to incur *false positives* (i.e., disclose points that are not part of the actual result) or *false negatives* (i.e., omit matching points from the result). It is possible to avoid false negatives altogether, by

ensuring that the mapping has the *contractiveness* property (i.e., normalized distances in the mapped space are smaller than in the original space). Mapping techniques may require the existence of a non-colluding third party [14].

Mapping techniques are efficient, but existing solutions suffer from privacy vulnerabilities when the adversary has background knowledge about the distribution of the data points in their domains. Note that, such attacks are particularly a concern in the case of spatial datasets, since the adversary can use publicly available maps to learn information about the possible placement of data points. Consider the example of Figure 1(a), corresponding to a spatial dataset. Note that, the distribution of data points is typical of an urban area, with a dense region in the bottom right corner (e.g., downtown), and an outlier point o (e.g. a remote suburb). According to the private matching method presented in [14], each point in the two-dimensional dataset is mapped to a multi-dimensional space using the SparseMap [7] embedding. Specifically, a number of k random reference point sets are chosen first. In the example, $k = 2$ reference point sets are agreed upon by the two parties: $S_1 = \{u\}$ and $S_2 = \{v\}$ (for simplicity, we assume one point per set). Assume that u and v fall within the dataspace region with a high density of points. Next, each data point p is embedded onto a new space, based on the distances from p to the reference sets (measured as the minimum Euclidean distance d_E from p to any point in the set). The embedded value of data point o (where F denotes the mapping) will be

$$F(o) = \{d_E(o, u), d_E(o, v)\} = \{9, 8\}$$

Similarly, the mapping of point q in the dense area will be $F(q) = \{1, 1\}$. Furthermore, all points in the dense region are likely to have an embedding similar in value with $F(q)$.

In the private matching protocol, the embedded data points are considered to be safe for disclosure, and are sent for matching to a third party (TP) who does not know the reference points (which act as a secret key). The TP determines pairs of points that satisfy the matching threshold, and returns *only* these pairs to both A and B. Note that, in general, matching mapped points may not be fully accurate, in the sense that false positives (points that do not actually match in the original space) or false negatives (points that are not included in the result although they match) may occur. The TP is assumed to be semi-honest, in the sense that it performs the matching on top of the mapped data correctly, but it may attempt to infer the actual locations of the data points. The TP is likely to have background knowledge about the distribution of points in the dataspace (e.g., the fact that the remote suburb is very sparse). Therefore, the TP can infer that a value of F equal to $\{9, 8\}$, which is significantly different than the $\{1, 1\}$ mapping of all other points, is likely to correspond to an outlier. Thus, an adversary can learn that point o corresponds to an entity (e.g., customer) situated in the suburb, compromising privacy.

The solution in [17] suffers from a similar drawback. In addition to [14], each party further maps its SparseMap-embedded points to the complex plane. In particular, the SparseMap image for point p

$$F(p) = V = \{v_1, \cdots, v_k\}$$

is further mapped to complex point

$$c(V) = \sum_{j=0}^{k-1} v_j e^{\frac{j \times 2\pi \mathbf{i}}{k}}$$

where $\mathbf{i} = \sqrt{-1}$ signifies the imaginary component. The

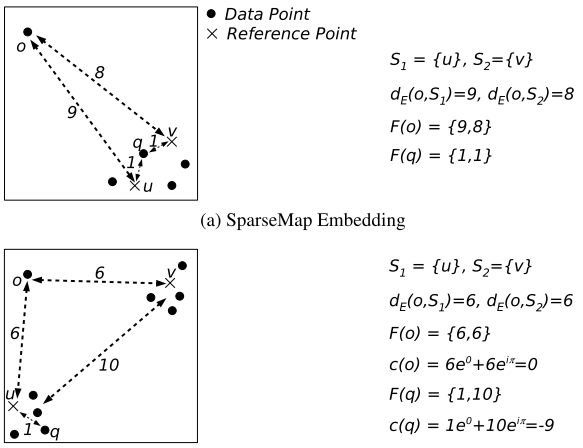


Figure 1: Existing Approaches Compromise Confidentiality of Outliers

work in [17] assumes that if the components of V are not disclosed separately, but only $c(V)$ is released instead, then there is no need for TP, and parties A and B can directly exchange their mapped datasets. This approach compromises the confidentiality of outliers as well. Consider the example of Figure 1(b): this time there are two clusters of data points, and the reference points are u and v. If the outlier o is situated at roughly equal distance from the dense areas, then the two terms of the sum $c(F(o))$ will correspond to vectors in the complex plane that have equal modulus and opposite orientations (i.e., 0 and π). Therefore, they will cancel each other out, and the resulting mapped value will be close to the origin. On the other hand, for any other data point, the modulus of the resulting complex point will be roughly equal with the distance between the two clusters. Hence, an attacker can easily identify that a point mapped close to origin must be the outlier o. Note that, the reverse-engineering remains successful even if a third party is used.

1.1 Problem Formulation and Contributions

Consider parties A and B and their respective spatial datasets D_A and D_B. Given a distance threshold δ expressed with respect to the Euclidean distance d_E, the two parties want to determine the result of the join set

$$D_A \bowtie_\delta D_B = \{(p, q) \mid p \in D_A \wedge q \in D_B \wedge d_E(p, q) < \delta\}$$

while minimizing the amount of disclosed data points that are not part of the join result. Since none of the points of the other party are known in advance, both A and B must include all of their points in the join query processing. To prevent disclosure, they first transform their points. Denote by $D_A \bowtie_\delta D_B$ the *actual* result of the join, and by R the result obtained using the transformed data[1]. We identify several important properties that characterize solutions for private matching:

- *Robustness against background-knowledge attacks.* Given a transformed data point, an adversary must not be able to identify its original location. The example in Figure 1 has shown that existing techniques fail to protect the location of outliers.

[1]We emphasize that R includes points that are disclosed in plaintext format due to false positives, and does not account for locations that are reversed-engineered

- *Precision.* The joining process must not disclose any original points that are not part of the result. Formally, precision is defined as
$$\frac{|(D_A \bowtie_\delta D_B) \cap R|}{|R|}$$
- *Recall.* The joining procedure must return the complete result. Recall measures the number of missed results, i.e.,
$$\frac{|(D_A \bowtie_\delta D_B) \cap R|}{|D_A \bowtie_\delta D_B|}$$
- *Overestimation Ratio (OR).* For every pair $(p, q) \in D_A \bowtie_\delta D_B$, the distance $d_E(p, q) < \delta$. If a method obtains false positives, then in addition to the number of false positives, it is also important to determine how far apart these points are compared to δ. For instance, if $d_E(p, q) = 110$ meters and the threshold is $\delta = 100$ meters, the disclosure is more acceptable than if $d_E(p, q) = 1000$ meters. OR is defined as
$$OR = \frac{\max_{(p,q)\in R} d_E(p, q)}{\delta}$$

An ideal technique for matching must be robust against reverse-engineering, and must achieve 100% precision and recall (which implies $OR < 1$).

In this paper, we propose a geometric transformation method for private matching of spatial datasets which is robust to background knowledge attacks. Specifically:

(*i*) We propose a novel two-level grid geometric transformation that is robust against adversaries with background knowledge about data points distribution. Similar to [14], the proposed method requires a semi-honest third party, and achieves 100% recall. However, it clearly outperforms [14] in terms of precision and overestimation ratio.

(*ii*) We explore two alternatives for measuring distance in the transformed space, namely the L_2 and L_∞ metrics, that offer an interesting trade-off between privacy and the amount of resulting false positives.

(*iii*) We perform an extensive experimental evaluation showing that the two-level grid solution clearly outperforms existing work in terms of precision and overestimation ratio, and is faster in terms of execution time overhead.

The rest of the paper is organized as follows: in Section 2, we survey related work. Section 3 gives an overview of our approach. Section 4 presents the details of the private matching protocol. We present the results of our experimental evaluation in Section 5, and conclude in Section 6.

2. RELATED WORK

The problem of spatial joins has been extensively studied in the database literature. A comprehensive review of such methods can be found in [15]. The performance of spatial joins can benefit from the existence of spatial indices (e.g., R-tree) on top of the data [4]. Other prominent techniques to reduce join complexity are hash-join [10] and sweeping [3] algorithms. All these techniques assume that the data are public, hence they do not protect data confidentiality.

Private matching techniques that rely on data mapping employ geometrical or algebraic transformations to hide the data values [14, 17]. When the adversary has access to background knowledge, such techniques compromise data privacy, as shown in the motivating example of Figure 1. Another solution for private matching of relational data using *sanitization* has been proposed in [8]. The data is transformed to satisfy k-anonymity [13], which requires that each attribute combination must be shared by at least k records. The technique can be easily adapted to spatial data, and results to generalization of locations, i.e., the release of rectangular regions instead of point locations. The drawback is that, even if exact coordinates are not disclosed, a lot of information is learnt by the adversary by observing the released 2D rectangles.

Symbol	Description
w	size of major-grid cell
k	granularity of minor-grid ($k \times k$)
δ	matching threshold
r	number of rotations
$u_i = (u_i^x, u_i^y), (1 \le i \le r)$	rotation pivots
$\alpha_i, (1 \le i \le r)$	rotation angles
$p' = (p'.c_1, \ldots, p'.c_r)$	mapping of data point p

Table 1: Summary of Notations

The work in [1] introduced a private join method that uses a secure CPU at a *trusted* third party (TTP). The TTP receives the data points in encrypted form from the data owners, and performs the join processing within the secure co-processor. This solution is efficient and effective, but it requires expensive specialized hardware. Several approaches for set intersection (i.e., spatial join with $\delta = 0$) using cryptographic techniques are reviewed in [9]. The work in [2] also presents a technique for set intersection that relies on commutative encryption. However, set intersection does not support distance-based matching with thresholds $\delta > 0$.

SMC techniques [6] allow private evaluation of any generic function $\mathcal{F}$ defined on inputs owned by distinct parties. SMC protocols are accurate and secure, but they require complex implementations, due to the need to express function $\mathcal{F}$ as a binary circuit. Furthermore, the computation and communication complexity of SMC are prohibitive in practice.

Finally, the privacy of location data has also been considered in the context of private location-based queries. The purpose is to hide the exact identity and/or location of mobile users asking for nearby points of interest. Several methods have been proposed that use k-anonymous location generalization [11] or encryption [5]. This setting is different from that of private spatial joins, since only the user locations are hidden, whereas the locations of points of interest are not. For private joins, the data points stored at both parties must be protected.

3. SOLUTION OVERVIEW

Similar to the work in [14, 17] we map data points from 2D coordinates to a multi-dimensional space; however, our mapping is specifically designed to protect against adversaries with background knowledge. The key idea behind our transformation is the use of a two-level grid. Each level-1 regular grid splits the data space into a set of *major* cells, and each major cell is further split according to a level-2 grid into a set of *minor* cells. For the mapping we use multiple instances of the two-level grid, where each instance represents a rotation of the grid by a random angle around a randomly-chosen pivot.

We motivate our choice for a two-level grid design by looking at two attack venues that are representative for adversaries with background knowledge. Specifically, we focus on *distance-based* and *placement-based* attacks. The example of Figure 1 illustrated how distance-based attacks work: the

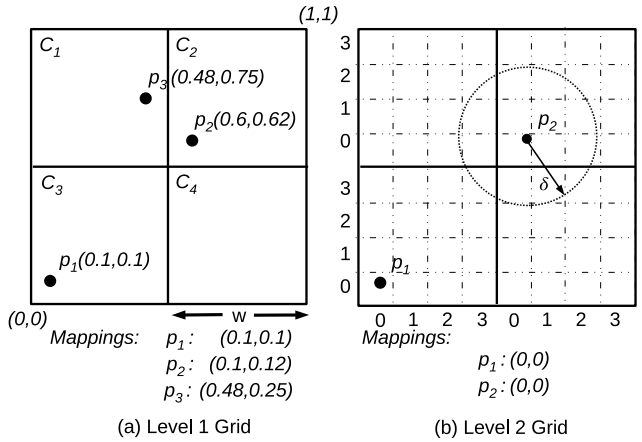


Figure 2: Two-level Grid Design

main observation is that if the geometric transformation of datasets preserves large distances between points, then an attacker can identify certain patterns among the mapped points that allow the isolation of outliers. The proposed first-level grid illustrated in Figure 2(a) partitions the data space into a number of major cells of size $w \times w$ each, and the coordinates of each point are considered with respect to the lower-left corner of the enclosing major cell (e.g., the mapping of p_2 corresponds to its coordinates within cell C_2). The major grid reduces the distance between each two points to a maximum of $w\sqrt{2}$, effectively eliminating the notion of outlier from the attacker's view of the data, hence thwarting distance-based attacks. For instance, although points p_1 and p_2 are situated far apart from each other, the distance between their mapped values is small.

While the major-grid split protects against distance-based attacks, it is still vulnerable to another type of attacks, which we call placement-based attacks. Revisiting the example in Figure 2(a), consider that the attacker has knowledge of the existence and exact coordinates (i.e., placement) of point p_1. Furthermore, assume that p_1 is close to the lower-left corner of the dataspace, hence the attacker can infer with high probability that p_1 is enclosed by the lower-left major-grid cell C_3. In this case, the mapped coordinates of p_1 coincide with its actual coordinates. If the coordinates p_1^x and p_1^y are expressed with high precision, then it is possible that no other point in the dataset has the mapped value equal to that of p_1 in at least one of the x and y coordinates. For instance, neither p_2 nor p_3 have 0.1 in the mapped y value. Therefore, the adversary learns that p_1 must belong to the dataset.

To protect against placement attacks, the exact coordinates of data points should not be disclosed. Instead, we perform a discretization of the data space by dividing each major-grid cell according to a $k \times k$ minor-grid. Points within the same minor cell are mapped to the same coordinates, representing the position (row and column) of the minor cell inside the major cell. A similar division structure is repeated for all the major cells, consequently all the points enclosed within the same row and column (belonging to the *same* or *different* major cells) share the same combination of coordinates. For instance, in Figure 2(b) a 4×4 minor-grid ($k = 4$) is used, and both p_1 and p_2 are mapped to value $(0, 0)$, which corresponds to the coordinates of the lower-left minor-grid cell in each of the corresponding major cells.

Note that, the major/minor grid transformation protects the location of points, but it also has a negative impact in the precision of evaluating matching pairs. For instance, points p_1 and p_2 that have the same mapped value in Figure 2(b) will be considered as matching, although the Euclidean distance between them is larger than threshold δ (only information about the minor cells, and not the major cells, is employed in the matching decision). This concept can be visualized in Figure 3, that illustrates the cells matching with point p for $k = 6$ and two different thresholds. First, p is mapped to its minor cell coordinates, i.e., $p(5, 4)$. It is straightforward to see that, for a certain threshold δ, the matching cells are the cells situated at a Euclidean distance $< \delta$ of *any* point within cell $(5, 4)$. This spatial constraint is represented by the Minkowski sum of cell $(5, 4)$ and a circle with $radius = \delta$. The Minkowski sums for two choices of the matching threshold ($\delta = 1$ and $\delta = 1.7$) are shown. For each of them, the shadowed areas mark the cells contained or intersected by the Minkowski sum, which represent the matching cells. Due to the replication of the minor cell coordinates in each major cell, the group of matching cells appears at the same relative position in each major cell.

In order to facilitate pruning of non-matching pairs of points based on their mapped values, we employ several distinct instances of the two-level grid, obtained through rotation by a random angle around a randomly chosen pivot point. Figure 4 shows an example with two grids, $k = 6$ (for simplicity, the pivot point is not shown). The mapped value of each point consists of two pairs of coordinates, one pair for each grid (the first two coordinates correspond to the grid with zero-rotation angle). To facilitate the presentation, we consider that the join condition requires that two points match if the L_∞ distance between their mapped coordinates is at most $\delta = 1$ in each grid (in Section 4.3 we discuss matching with the Euclidean distance). There are four matching pairs of points according to $Grid_1$, but only three according to $Grid_2$. It is straightforward to prove that the two-level grid mapping is contractive with respect to the L_∞ metric, therefore a pair of points can match in the original data space only if *all* their mappings (with respect to each individual grid) match. By using several distinct rotated grids we can improve the matching precision.

The parameter notations are summarized in Table 1. The choice of parameters is important with respect to both privacy and matching precision. The primary objective of the two-level grid structure is to prevent disclosure of point locations through distance- and placement-based attacks. However, due to the fact that the distances are transformed,

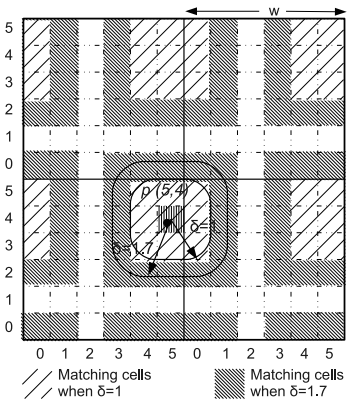


Figure 3: Minkowski sum and matching cells

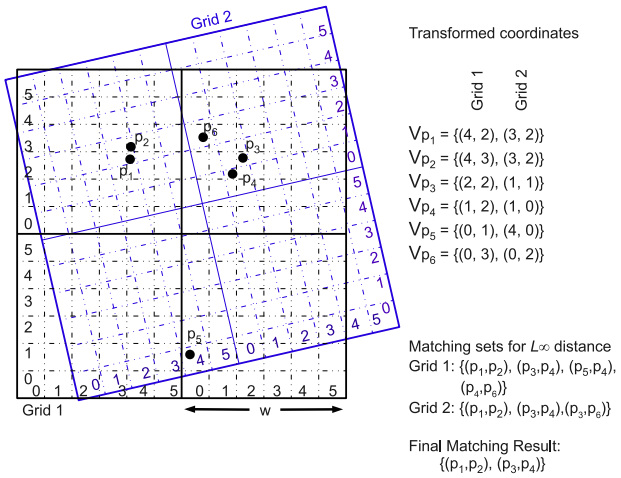


Figure 4: Matching with Rotated Grid Instances, $\delta = 1$ (L_∞ metric)

matching precision decreases. The choice of w is particularly important, and it depends on the value of the matching threshold δ. For instance, if $w \leq \delta\sqrt{2}$, no pruning is possible with the two-level grid. However, in general, threshold distances are small compared to the dataspace size (for instance, in a 50×50 km city, it may be interesting to find pairs of points situated at most 500 meters apart from each other, i.e., 1% of the dataspace). Therefore, it is possible to find w values that are several times larger than δ, but still considerably smaller than the data space size. We explore in detail the effect of grid parameters on precision in the experimental evaluation.

4. PRIVATE MATCHING PROTOCOL

In this section, we present the details of the private-matching protocol. We describe the system architecture in Section 4.1, and we show how the two parties map their datasets in Section 4.2. Next, we focus on match processing: in Section 4.3 we describe the join protocol that relies on Euclidean distance in the transformed space, whereas in Section 4.4 we show how to improve privacy by decoupling x and y coordinates and using L_∞ distance. Finally, Section 4.5 provides a security discussion.

4.1 System Architecture

The system architecture consists of three entities: the data owners A and B, and the third party TP. The TP may try to infer the actual placement of points of A and B, but we assume that it will not be able to collude with either A or B. We assume a semi-honest model, where all parties follow the protocol, but they may try to infer additional information.

The private matching process with geometrical transformations consists of four steps, and is illustrated in Figure 5:

(i) parties A and B agree on the transformation parameters w, k, r, α_i and u_i ($1 \leq i \leq r$), as well as matching threshold δ (step 1).

(ii) parties A and B transform their datasets according to the chosen parameter set (step 2) and send their mapped data to TP, as well as k and the matching threshold for the transformed space (see Section 4.3) (step 2′).

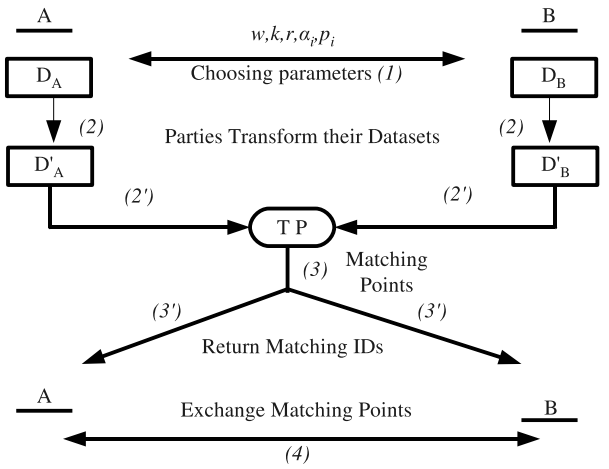


Figure 5: System Architecture

(iii) TP processes the spatial join with respect to the mapped datasets (step 3) and returns the matching point identifiers to parties A and B (step 3′).

(iv) parties A and B exchange the actual points corresponding to the matching identifiers (step 4).

4.2 Data Mapping in the Two Parties

The two parties need to transform their data points into vectors that will be processed by the TP in the next step. The transformed vector of a point consists of a list of $2 \times r$ integers corresponding to the coordinates of the minor cell enclosing the point in each of the r grid instances. Each instance corresponds to a rotation of the original data space with angle α_i around a pivot $u_i = (u_i^x, u_i^y)$, where $1 \leq i \leq r$. The resulting mapped value p' is denoted by

$$p' = \left(p'.c_1, p'.c_2, \ldots, p'.c_r\right),\ c_i = (c_i^x, c_i^y), c_i^{x,y} = 0..k-1 \quad (1)$$

The complete mapping procedure executed by each party is described in the following pseudocode:

Dataset Mapping
Input: dataset D, w, k, r, α_i, $u_i = (u_i^x, u_i^y)$
Output: transformed dataset D'
1. **for** $i = 1$ **to** r **do**
   /* create transformation matrices */
2. $R_i = \begin{bmatrix} \cos\alpha_i & \sin\alpha_i & 1 \\ -\sin\alpha_i & \cos\alpha_i & 1 \\ 0 & 0 & 1 \end{bmatrix}$, $T_i = \begin{bmatrix} 1 & 0 & -u_i^x \\ 0 & 1 & -u_i^y \\ 0 & 0 & 1 \end{bmatrix}$
3. **forall** $p \in D$ /* compute mapped vector p' for each p */
4. $p' = \emptyset$
5. **for** $i = 1$ **to** r **do**
6. $[p_i^x\ p_i^y\ 1]^T = T_i^{-1} \times R_i \times T_i \times [p^x\ p^y\ 1]^T$
7. $c_i^x = \lfloor (p_i^x - w \cdot \lfloor p_i^x / w \rfloor) \cdot k / w \rfloor$ /* x-minor cell */
8. $c_i^y = \lfloor (p_i^y - w \cdot \lfloor p_i^y / w \rfloor) \cdot k / w \rfloor$ /* y-minor cell */
9. $p' = p' || (c_i^x, c_i^y)$
10. $D' = D' \cup p'$

Figure 6: Dataset Transformation

4.3 Query Processing at the TTP

The TP determines the matching pairs of points in the mapped dataspace, based on the point mappings sent by parties A and B. Note that, in the mapped space, the distance measurement unit is the side length of a minor cell (w/k), hence the matching threshold becomes

$$\delta' = \delta \times k \div w. \quad (2)$$

This value is sent to the TP by either A or B, together with their transformed points. Note that neither the original threshold δ, nor the window size w need to be disclosed to the third party. Although the value of k is required for the distance computation, this disclosure does not lead to any privacy leakage as this value can be learnt by looking at the coordinates c_i^x, c_i^y of the transformed vectors.

The matching function for every candidate pair of vectors $(p'_A \in D'_A, p'_B \in D'_B)$ computes the distance *cdist* between the cells of p'_A and p'_B for each rotated grid instance, and compares it to δ'. The pair (p'_A, p'_B) represents a match only if $cdist < \delta', \forall i = 1..r$, i.e., for all rotated grid instances. Figure 7 shows the algorithm executed by the TP.

Matching
Input: transformed datasets D'_A, D'_B, matching threshold δ'
Output: set of matching pairs S
1. $S = \emptyset$
2. **forall** $p'_A \in D'_A$
3. **forall** $p'_B \in D'_B$
4. **for** $i = 1$ **to** r **do** /* do matching for every rotation */
5. $c_A = (p'_A.c_i)$ /* minor cell of p'_A for rotation r */
6. $c_B = (p'_B.c_i)$ /* minor cell of p'_B for rotation r */
7. **if** $cdist(c_A, c_B) \geq \delta'$ /*cell distance*/
8. **break** /* discard pair */
9. **if** $i = r$ /* if we reached the last rotation */
10. $S = S \cup (p'_A, p'_B)$ /* add pair to the result */

Figure 7: Matching in the third party

The function $cdist(c_A, c_B)$ computes the distance between the two minor cells c_A and c_B, defined as the Euclidean distance between the two closest points of c_A and c_B. Note that two cases must be considered: *(i)* c_A and c_B belong to the same major cell, or *(ii)* they belong to distinct major cells. For the first case, *cdist* is given by

$$cdist(c_A, c_B) = \sqrt{\Delta_x^2 + \Delta_y^2} \quad (3)$$

where

$$\Delta_x = \begin{cases} 0, & c_A^x = c_B^x \\ |c_A^x - c_B^x| - 1, & otherwise \end{cases}$$

and similarly for the y coordinate. Figure 8 shows how this distance is computed: in major cell C_0, the distance between p_A and p_B is given by $d_0 = \sqrt{2^2 + 3^2} = \sqrt{13}$.

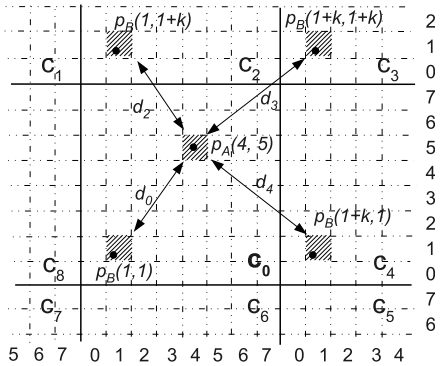


Figure 8: Euclidean distance calculation between grid cells

For the second case (when the points belong to different major cells) we must consider the minimum possible distance between them. In particular, since we are interested in the minimum distance, we only need to consider combinations of major cells that are neighbors (i.e., major cells C_0 to C_8 in Figure 8). Point p_A is marked only in the major grid cell C_0, whereas the placement of p_B is illustrated for multiple major cells. Formally[2],

$$\begin{aligned} cdist(c_A, c_B) &= \min_{i,j} cdist(c_A, c'), \\ where \quad & c' = (c_B^x + i, c_B^y + j), \\ & i \in \{-k, 0, k\}, j \in \{-k, 0, k\} \end{aligned} \quad (4)$$

Note that this definition of distance is symmetric, i.e., $cdist(c_A, c_B) = cdist(c_B, c_A)$. In addition, due to symmetry across major cells, we can compute $cdist(c_A, c_B)$ by translating the two minor cells of points p_A and p_B such that c_A is moved to the origin (cell $0, 0$) and c_B is moved to the cell $(|c_A^x - c_B^x|, |c_A^y - c_B^y|)$. This property is shown in Figure 9(a), where the distance between cell $c_A(2, 6)$ and $c_B(6, 4)$ is equivalent to the distance between $c''_A(0, 0)$ and $c''_B(4, 2)$. Then, the computation in Eq.(4) only needs to be performed for the *left*, *bottom-left*, *bottom* and *center* major cells (see Figure 9(b)). This optimization reduces the number of distance evaluations from nine to four.

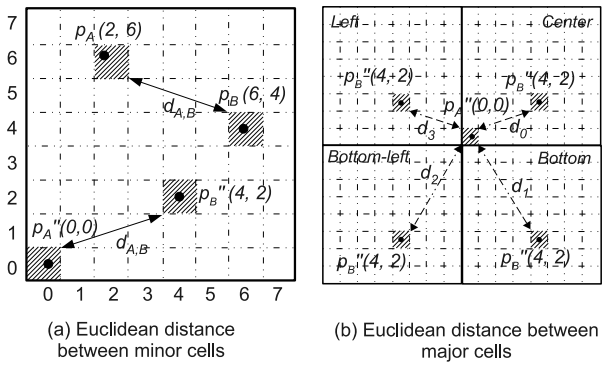


Figure 9: Optimization of distance calculation

4.4 Additional Privacy: Separating x and y coordinates

The method described in the previous section requires the disclosure of minor cells coordinate pairs (c_x, c_y) for each point, which may help an adversary to learn the exact placement of the point within its major cell. Since the actual major cell is not disclosed, the attacker does not gain significant knowledge of the position of the point in the data space, especially if w is small. Nevertheless, we can improve privacy by breaking the correlation between the c_x and c_y cell coordinates of each mapped point, as discussed next.

Consider the Chebyshev distance (L_∞) between a pair of cells, which is defined as the *maximum* of their differences along any of the coordinates x and y. If a pair is a match, then the distance must be below a threshold for *both* coordinates, and therefore the evaluation can be done separately for each dimension. Furthermore, the Chebyshev distance is a lower bound of the Euclidean distance and therefore preserves the contractiveness property (i.e., no false negatives can occur).

When using Chebyshev distance, the transformed match-

[2]We denote the placement of cell c' with coordinates outside the $0..k-1$ values in order to illustrate how distances are calculated. We emphasize that the TP only sees $0..k-1$ cell identifiers.

ing threshold changes to:

$$\delta_c = \lceil \delta \times k \div w \rceil = \lceil \delta' \rceil, \quad (5)$$

which is equivalent to aligning δ' to a multiple of minor grid cell side length. The utilization of a coarser distance approximation may lead to decreased precision. However, as we show in the experiments, the precision and overestimation ratio are not significantly affected.

The distance function between minor cells changes compared to the case of Euclidean distance. Specifically, if two minor cells are placed in the same major cell, we have

$$cdist(c_A, c_B) = \max(|c_A^x - c_B^x|, |c_A^y - c_B^y|) \quad (6)$$

The distance evaluation for all possible major cell placements is:

$$cdist^x(c_A, c_B) = min(|c_A^x - c_B^x|, k - |c_A^x - c_B^x|) \quad (7)$$

$$cdist^y(c_A, c_B) = min(|c_A^y - c_B^y|, k - |c_A^y - c_B^y|) \quad (8)$$

The term $k - |c_A^x - c_B^x|$ (or $k - |c_A^y - c_B^y|$ for the y coordinate) reflects the case where the coordinates belong to neighbor major cells. Recall that, the minimum distance between any pair of cells is always located in one of the following cases: *(i)* the cells belong to the same major cell, or *(ii)* the cells belong to neighbor major cells. For the latter, the distance between two neighbor cells in one dimension is at most k, therefore the difference for the coordinates becomes $k - |c_A^y - c_B^y|$. Figure 10 illustrates this concept for the distance between $p_A(4, 6)$ and $p_B(1, 1)$ for the two coordinates separately, with $k = 8$. For instance, for the y coordinate $cdist^y(c_A, c_B) = min(|6 - 1|, 8 - |6 - 1|) = min(5, 3) = 3$, and similarly for the x coordinate.

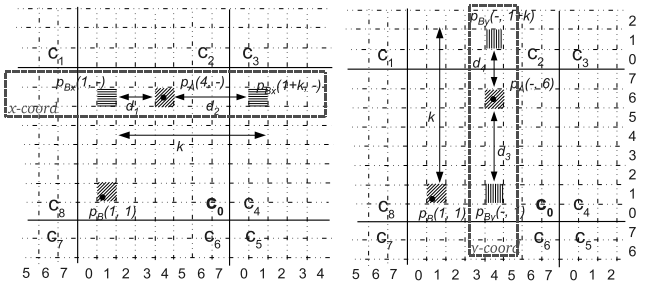
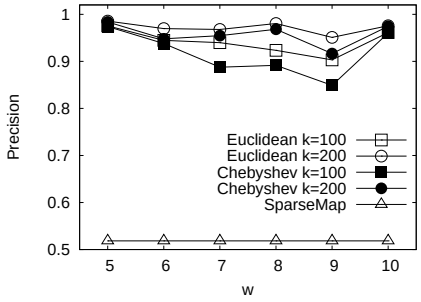


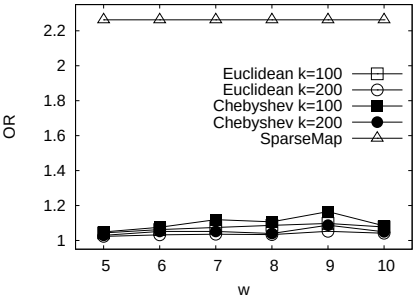
Figure 10: Chebyshev distance calculation across distinct major cells

The matching algorithm executed in the third party is similar to the one in Figure 7, except that the values retrieved in lines $5 - 6$ are single coordinates (as opposed to cell coordinates). Therefore, the distance computation in line 7 is modified to account for Chebyshev distance, and the *for* loop in line 4 is executed for twice as many iterations, one per each separate coordinate x and y.

Note that, since the two components of the Chebyshev distance (x and y) can be evaluated independently, the TP no longer requires a particular order of the coordinates in the multi-dimensional vector representing a point mapping (Eq.(1)). Therefore, parties A and B can agree on a random permutation used by both when arranging the cell coordinates of the same point for distinct grid instances. This prevents the TP from correlating x and y coordinates of a point within the same rotated grid.



(a) Precision



(b) Overestimation Ratio

Figure 11: Varying w, $\delta = 1\%$, $dbSize = 15\%$, $r = 20$

4.5 Security Discussion

As stated in Section 1.1, the objective of private joins is to prevent an adversary from recovering the original coordinates of a point, given its mapping. Our proposed two-level grid compresses all distances in the original space to less than $w\sqrt{2}$ (the diagonal of a level-1 cell), therefore outliers cannot be identified in the mapped data. Furthermore, the use of level-2 cells hides the exact point coordinates and protects against placement-based attacks.

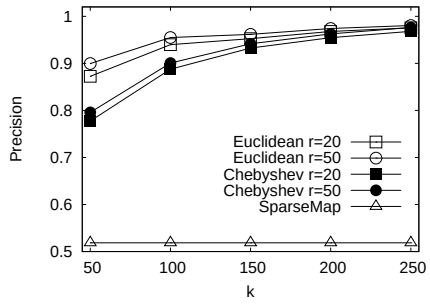
Assume that an attacker attempts to find whether one known point is in the data set of one of the parties. Consider the pseudocode in Figure 6: an attacker may feed the coordinates (p_j^x, p_j^y) of the known point into the equation in line 6, and set up a system of $2r$ equations, in which there are $2r + 3$ unknowns (including α_i, u_i^x, u_i^y). Such a system has infinitely many solutions, and the attacker is not able to determine the mapped point values, hence to learn if the known point is contained in the data.

5. EXPERIMENTAL EVALUATION

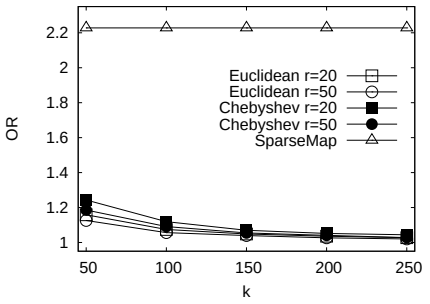
We implemented prototypes of the proposed geometric transformations (labeled *Euclidean* and *Chebyshev*). Our experimental workload consists of the Sequoia[3] data set with $65K$ records. We generate the sets D_A and D_B of the two parties by random sampling. We consider threshold values δ between 0.5% and 10% of the dataspace, and we vary the data size of each party between 6500 (10%) and $32,000$ (50%) of the Sequoia set. The experiments were run on a P4 3.0GHz machine with 1GB of RAM running WinXP.

In Section 5.1 we evaluate the precision and overestimation ratio of the transformations in comparison with the method in [14] labeled *SparseMap*. We measure the robustness to attacks of the two-level grid scheme in Section 5.2, and we present the running time performance of the studied methods in Section 5.3.

[3]Available at http://www.rtreeportal.org

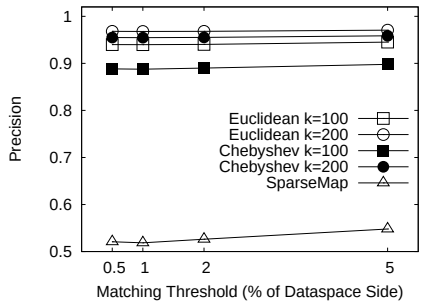


(a) Precision

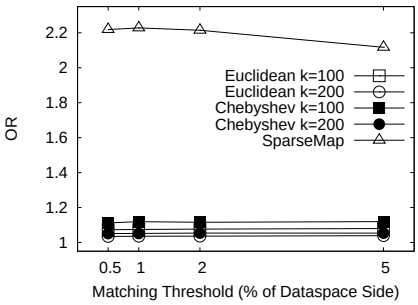


(b) Overestimation Ratio

Figure 12: Varying k, $\delta = 1\%$, $w = 7\delta$, $dbSize = 15\%$



(a) Precision



(b) Overestimation Ratio

Figure 13: Varying δ, $w = 7\delta$, $dbSize = 15\%$, $r = 20$

5.1 Precision and Overestimation Ratio

First, we measure the precision and OR[4] when varying the granularity of the major grid (i.e., major cell extent w). We choose the values of w as a multiple of matching threshold δ, which in this experiment is fixed to 1% of the dataspace side (recall from Section 3 that precision is influenced by the ratio between w and δ). Figure 11 shows that the precision of *Euclidean* is always above 90%. As expected, *Chebyshev* achieves slightly lower precision, but still always above 83%, as it trades off precision for better privacy. Nevertheless, its precision is much higher than $SparseMap$. Note that, this result is obtained for w values that are small multiples of δ, which means that pruning with the multiple rotated grid instances method is effective even if major grid cells have small extent (hence provide good location hiding). Our techniques outperform $SparseMap$ in terms of OR as well. The maximum distance between any pair of points returned as result is at most 1.17δ, which is a reasonably accurate approximation of the actual threshold.

In Figure 12 we show the effect of minor grid granularity. As expected, a finer granularity of the minor grid (i.e., larger k) leads to more precise matching, due to the higher accuracy in evaluating the distance between cells. Even for the lower $k = 50$ value, our techniques clearly outperform $SparseMap$ in terms of precision and OR. In practice, values of $k = 100$ or above achieve very good precision (more than 88% for *Chebyshev* and 90% for *Euclidean*).

In Figure 13 we vary threshold δ and set $w = 7\delta$. Note that, neither the precision nor the OR of our techniques are significantly influenced by δ, since these are decided by the relative ratio between w and δ. On the other hand, the precision of $SparseMap$ deteriorates for lower values of δ, as the method does not handle well accurate distance evaluations. A similar relative performance can be observed for the OR metric.

In Figure 14 we vary the size of the datasets of the two parties (we assume that both have the same number of data points). The precision and OR of our methods are not influenced by dataset size (and density). The reason for this behavior is the use of a regular grid partitioning, as the performance is mainly dictated by the grid parameters. In contrast, $SparseMap$ is less robust to data distribution, and shows considerable variations as data density changes.

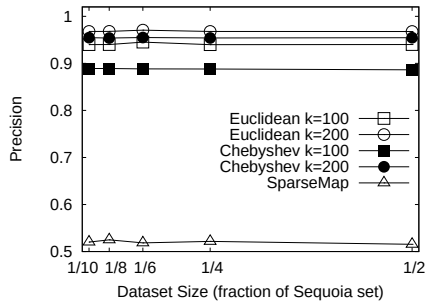
In Figure 15, we evaluate the effect of the number of grid instances r. As expected, a higher r increases the precision and reduces the OR, due to increased effectiveness of pruning: points that are far apart are likely to be classified as non-matching according to at least one of the grid instances. However, even for low r values (e.g., $r = 10$), the precision of our methods is good, therefore the communication cost can be maintained at a low level (recall that each party sends to the TP two cell coordinates for each grid instance).

5.2 Robustness to Attacks

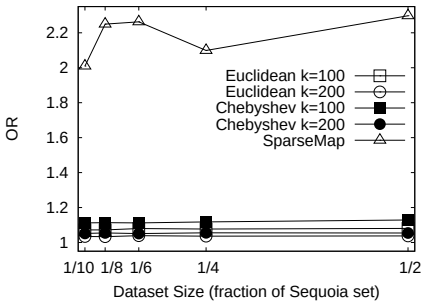
In this section, we evaluate empirically the robustness to attacks of the proposed geometric transformations. First, we consider a *distance-based attack* in which an adversary attempts to infer whether a certain location is present in the transformed dataset based on the distances from the candidate location to other points. Such means of identification are referred to as point *signatures* in [16]. Next, we perform a *cell occupancy analysis* which quantifies the ability of an attacker to distinguish among mapped points based on their coordinates. The objective of the analysis is to provide a robustness measure with respect to worst-case attack scenarios.

Distance-based Attack. As discussed in Section 1, distance-based attacks are particularly effective against outlier points. We consider that an adversary has knowledge about the data distribution, and creates an *attack set Att* consisting of points situated in sparse areas (i.e., outlier

[4]Recall that, both our method as well as [14] do not incur false negatives, so recall is always 100%.

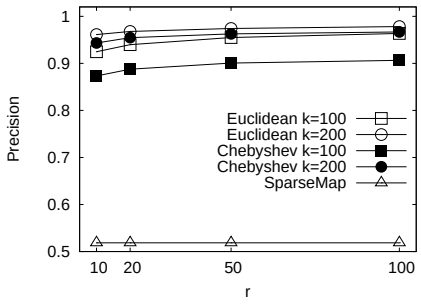


(a) Precision

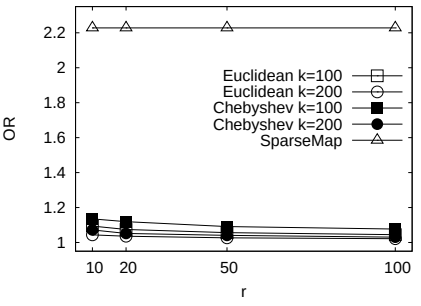


(b) Overestimation Ratio

Figure 14: Varying Data Size, $\delta = 1\%$, $w = 7\delta$, $r = 20$



(a) Precision



(b) Overestimation Ratio

Figure 15: Varying r, $\delta = 1\%$, $w = 7\delta$, $dbSize = 15\%$

candidates). The adversary's objective is to determine if the points in the attack set are present in the transformed dataset. To obtain Att, we sort the points in the Sequoia set in decreasing order of their distance to their n^{th} nearest neighbor (NN) (we set $n = 5$ to detect co-located outliers as well). Att consists of the first points in the order following which the distance remains virtually unchanged (a similar outlier detection method is used in [12]). For the considered dataset, we obtained $|Att| = 20$. We use the same procedure for the transformed points, and we determine the set Att' containing the $|Att|$ points with the farthest distance to their 5-NN in the mapped space. We measure the success of the attack as $|Att \cap Att'|/|Att|$.

Table 2 shows the attack success rate for various parameter settings (the w values correspond to major grids with granularity ranging from 4×4 to 20×20). No dataset point is correctly guessed by the attacker, except in one case when the success of the attack is 5%. However, only a single point is correctly guessed (as $|Att| = 20$). Note that, such an event could occur as a result of a random guess, regardless of the distance between points. On the other hand, the $SparseMap$ method is severely compromised.

Cell Occupancy Analysis. We examine the cell occupancy level for grid cells in the mapped datasets. Intuitively, regardless of the details of the attack staged, an adversary can only distinguish among points placed in different cells, but no distinction can be made between points that fall within the same cell. Therefore, cell occupancy is a good indicator of robustness against general types of attacks. Specifically, if a mapping results to an uniform distribution of points in each cell, it is robust, whereas if certain cells have few points (in the worst case a single point per cell), then an attacker may gain the ability to identify individual points with certain spatial characteristics.

w		**0.25**	**0.2**	**0.16**	**0.1**	**0.05**
$k = 100$	$r = 20$	0	0	0	0	0
	$r = 50$	0	5%	0	0	0
$k = 200$	$r = 20$	0	0	0	0	0
	$r = 50$	0	0	0	0	0
$SparseMap$		95%				

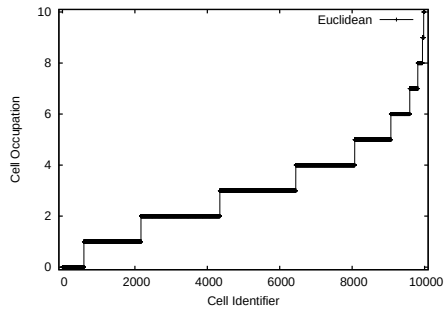
Table 2: Robustness against Distance-based Attacks

Figure 16(a) shows a sorted histogram with the cell occupancy for the Euclidean mapping, $k = 100$, over a random grid rotation sample. w is set to 7% of the dataspace. There are a total of $k \times k = 10,000$ (minor) cells, and a large number of them (roughly 1500) contain a single mapped point. On the other hand, the Chebyshev mapping is far better in spreading points uniformly over the grid cells. Figure 16(b) shows that the minimum number of points in a cell is 250, and the cell occupation levels follow a close-to-uniform distribution in the interval $250 - 350$. This experiment captures the advantages in terms of privacy of the Chebyshev mapping. Although the precision of Chebyshev is slightly worse than Euclidean, breaking the correlation between the horizontal and vertical coordinates of each grid cell brings significant benefits in terms of privacy.

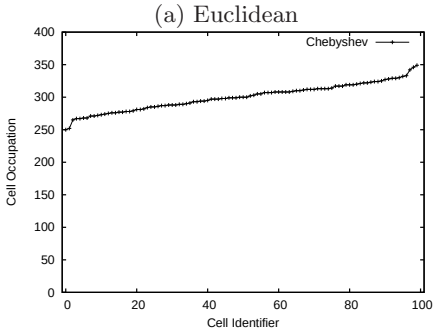
5.3 Execution Time

In this experiment, we evaluate the computational overhead of the proposed techniques for private spatial joins in comparison with the benchmark $SparseMap$. We set the distance threshold to $\delta = 1\%$, $w = 7\delta$ and we consider three settings for the number of rotations r: 20, 50 and 100. The execution time was observed to be very close for both $Euclidean$ and $Chebyshev$, so for brevity we only present the results for $Euclidean$.

Figure 17 shows the execution time for variable dataset size. The worst-case complexity of the join operation is quadratic in the dataset size, as can be observed from the behavior of $SparseMap$. Although our technique also shows a super-linear increase with dataset size, the execution time is much lower than that of $SparseMap$, due to the ability of effectively indexing data points based on their mapped cells. Specifically, by maintaining an inverted index for each cell storing the identifier of points mapped to that cell, we are



(a) Euclidean



(b) Chebyshev

Figure 16: Cell Occupancy Analysis, $w = 7\%$, $k = 100$

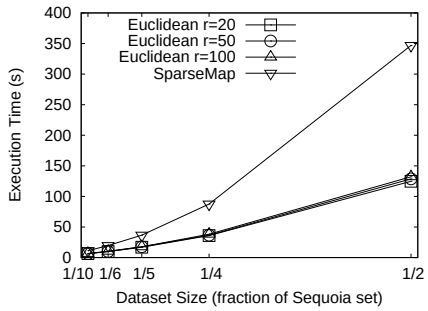


Figure 17: Execution Time

able to efficiently prune non-joining pairs without the need to evaluate the matching condition for individual points. In the worst case, our technique takes about two minutes to complete a join of two sets of $32,000$ points each, whereas $SparseMap$ requires about three times that amount.

Another interesting observation is that the execution time is virtually unchanged as r grows. Note that, in the worst case, the join complexity is linear to r, since the join condition may need to be evaluated for grid instances corresponding to all the rotations. However, in practice this is not the case, since a non-match in one grid instance means that the considered pair of points will not be part of the final result, and needs not be considered further. It turns out that the average number of rotated instances that must be evaluated before pruning a pair of points is small, therefore in practice large values of r can be used to improve precision without a significant impact on execution cost.

6. CONCLUSIONS

In this paper, we identified the limitations of existing private matching techniques in protecting spatial data against adversaries with background knowledge about the distribution of locations. We proposed private spatial join solutions based on geometric transformations that protect data confidentiality without sacrificing precision.

In future work, we plan to extend our semi-honest protocols to be robust against malicious adversaries. For instance, party A may misbehave and generate many fake points that join all points of B, therefore A learns the entire dataset of B. One direction towards solving this problem is to have each party commit to a set of encrypted points before executing the private join. Next, each party opens a small fraction of the committed points, randomly chosen by the other party, in order to prove the veracity of the points used in the join. In this setting, a challenging problem is to maximize the probability of detecting malicious behavior while minimizing the number of points disclosed in plain text. Malicious users can subsequently be penalized through policy-based or reputation-based mechanisms.

ACKNOWLEDGMENTS: The work reported in this paper has been partially supported by MURI award FA9550-08-1-0265 from the Air Force Office of Scientific Research.

7. REFERENCES

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

[2] R. Agrawal, A. Evfimievski, and R. Srikant. Information sharing across private databases. In *Proc. of ACM SIGMOD*, pages 86–97, 2003.

[3] L. Arge, O. Procopiuc, S. Ramaswamy, T. Suel, and J. Vitter. Scalable sweeping-based spatial join. In *Proc. of VLDB*, pages 570–581, 1998.

[4] T. Brinkhoff, H. Kriegel, and B. Seeger. Efficient processing of spatial joins using R-trees. In *Proc. of ACM SIGMOD*, pages 237–246, 1993.

[5] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, and K. L. Tan. Private Queries in Location Based Services: Anonymizers are not Necessary. In *SIGMOD*, 2008.

[6] O. Goldreich. The Foundations of Cryptography, Volume 2. *Cambridge University Press*, 2004.

[7] G. R. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):530–549, 2003.

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

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

[10] M.-L. Lo and C. V. Ravishankar. Spatial hash-joins. In *Proc. of ACM SIGMOD*, pages 247–258, 1996.

[11] M. F. Mokbel, C. Y. Chow, and W. G. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *Proc. of VLDB*, 2006.

[12] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proc. of ACM SIGMOD*, pages 427–438, 2000.

[13] P. Samarati. Protecting Respondents' Identities in Microdata Release. *IEEE TKDE*, 13(6):1010–1027, 2001.

[14] M. Scannapieco, I. Figotin, E. Bertino, and A. K. Elmagarmid. Privacy preserving schema and data matching. In *Proc. of ACM SIGMOD*, pages 653–664, 2007.

[15] S. Shekhar and S. Chawla. *Spatial Databases: A Tour*. Prentice Hall, first edition, 2003.

[16] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis. Secure k-NN Computation on Encrypted Databases. In *Proc. of ACM SIGMOD*, pages 139–152, 2009.

[17] M. Yakout, M. Atallah, and A. Elmagarmid. Efficient private record linkage. In *Proc. of ICDE*, pages 1283–1286, 2009.