Known Sample Attacks on Relation Preserving Data Transformations

Emre Kaplan, Mehmet Emre Gursoy, Mehmet Ercan Nergiz, and Yucel Saygin

Abstract—Many data mining applications such as clustering and k-NN search rely on distances and relations in the data. Thus, distance preserving transformations, which perturb the data but retain records' distances, have emerged as a prominent privacy protection method. In this paper, we present a novel attack on a generalized form of distance preserving transformations, called relation preserving transformations. Our attack exploits not the exact distances between data, but the relationships between the distances. We show that an attacker with few known samples (4 to 10) and direct access to relations can retrieve unknown data records with more than 95% precision. In addition, experiments demonstrate that simple methods of noise addition or perturbation are not sufficient to prevent our attack, as they decrease precision by only 10%.

Index Terms—Data mining, security and privacy, data transformations, known sample attacks, protection.

1 Introduction

Data privacy is a fundamental challenge, and various methods of data perturbation and obfuscation have been proposed to overcome this challenge. Among these, distance preserving transformations (DPTs) have gained significant attention due to their simplicity and ability to retain good accuracy for many data mining applications such as clustering and classification.

In this paper, we propose an attack on exact and approximate DPTs. Our attack relies on distance *relations* rather than actual distances themselves. That is, instead of asking "How close exactly are X and Y?", we ask "Is Y closer to X than Z?". The prior can be easily harmful since it may reveal Y's exact location, but the latter seems naive. A real-world application is through a location based social network: User X performs a proximity search by requesting a list of nearby users. Y is among the returned list of nearby users, but Z is not. Then, X infers that his distance to Y is smaller than his distance to Z. A DPT of this data enables the same inference to be made, since even though the data is transformed, distances between records are preserved. Furthermore, even if the distances are not exactly preserved, there can be an arbitrarily high probability of preserving distance relations, e.g., distance between X-Y is smaller than X-Z. We call such transformations that preserve distance relations (but not necessarily exact distances) relation preserving transformations (RPTs).

We show, through a novel attack, that the seemingly naive query involving distance relations can be harmful when used in a known sample attack. That is, if an attacker has a few known samples (that he is certain will be part of the private data) and can obtain distance relations concerning them and other unknown records in the data, then the attacker will be able to infer the contents of the unknown records with high accuracy and precision.

To better position our work within the literature, we first survey previous work on DPTs and attacks on DPTs. Then, we list our contributions and the primary characteristics of our attack.

Work on DPTs. Some data mining algorithms rely on distances between records rather than the records themselves. These algorithms work equally well without knowing the private data, but instead observing its distance matrix or transformed version (using a DPT). In [1], Chen and Liu show that the following are examples of such algorithms: k-NN classifiers, kernel methods, Support Vector Machines using polynomial, radial basis and neural network kernels, linear classifiers, and some clustering and regression models.

The three traditional techniques for DPTs are *translations*, *reflections* and *rotations* [2], [3]. Translations shift records a constant distance in parallel directions. Reflections map records to their mirror images in fixed-dimensional space. Rotations rotate all records by a fixed angle called the rotation angle. In more recent work, Huang et al. [4] present FISIP, in which they propose DPTs that preserve first order and second order sums and inner products of records. In [5], Giannella et al. argue that by tuning the parameters of random projection transformations, one can ensure arbitrarily high probabilities of distance preservation. They point to [6] for preliminary results in this direction.

Attacks on DPTs. Attacks on DPTs were first considered by Liu et al. [7], and later studied in [8], [9], [10] and [5]. We report the main differences between our work and these previous works. For a more general survey, we refer the reader to [11].

In [7], Liu et al. develop two attacks on DPTs, one where the attacker has a set of known samples, and one where the attacker has a set of known input-output pairs. In the latter, the attacker knows the true values of perturbed records, and this assumption is significantly stronger than what we assume in our work. We assume a known sample attack, which coincides with Liu et al.'s first attack. However, while their attack requires a significant number of known samples (e.g., 5% of the data, which can correspond to hundreds of samples) to be successful, we can achieve same rates of disclosure with only 4-6 samples. Thus, our attack is significantly more realistic and practical. In [9], Turgay

- *E. Kaplan and Y. Saygin are with the Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey. E-mail: {emrekaplan,ysaygin}@sabanciuniv.edu*
- *M. E. Gursoy is with the College of Computing, Georgia Institute of Technology, Atlanta, GA, 30332.E-mail: memregursoy@gatech.edu*
- *M. E. Nergiz is with Acadsoft Research, Gaziantep, Turkey. E-mail: nergiz@gmail.com*

For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TDSC.2017.2759732

et al. extend the attacks in [7] by assuming that the attacker has a distance matrix of the private data. They assume that the actual distribution of the data is known by the attacker, and develop attacks based on principal component analysis. In comparison, our attack works without knowledge of the data distribution.

In [8], Guo and Wu assume that the attacker has a set of known samples and aims to retrieve the remaining data using Independent Component Analysis (ICA). Their attack applies to arbitrary transformations, but their experiments imply that the attack requires approximately 500-1000 known samples to reach the precision of our attack. In [10], Chen et al. confirm that although DPTs (in particular, rotation and geometric perturbation) are useful for outsourced data mining, they are not resilient against ICA-based attacks. Therefore Chen et al. propose algorithms that compute ICA-resilient perturbations with additive noise, which meet a desired level of privacy guarantee and attack resilience.

Most recently, Giannella et al.'s attack in [5] achieves precision most similar to ours. However, their approach is probabilistic whereas ours is deterministic. That is, they identify records' original locations with certain confidence, whereas (without the presence of noise) we can identify locations always with 100% confidence. In addition, even though Giannella et al.'s attack as well as the attacks discussed above require DPTs, our attack runs on RPTs, which is a more relaxed and generalized set of transformations.

Contributions. In this paper, we propose an attack on RPTs. The highlights and distinctive features of our attack are as follows: (1) We generalize from DPTs to RPTs: RPTs only preserve the distance relationships while transforming the data, whereas DPTs preserve exact distances. We show that DPTs are a subset of RPTs, and therefore our attack directly applies to DPTs. (2) We base our attack solely on a set of known samples and relations. As such, the attacker need not even observe the transformed data or its distance matrix. Also, the attacker need no apriori knowledge of data distribution. (3) The attack is applicable to the distance matrix publication model [9] as well as the perturbed data publication model [8], since both models trivially leak distance relations. (4) We achieve high precision with very small and realistic known sample sizes, e.g., 2-6 known samples. Concretely, when attacking a target record with only 4 known samples, the attacker can reduce the data space by 96%, leaving only the remaining 4% as to where the target record could be located. Thus, our approach is more effective than many of the previous works. (5) We propose extensions, such as space discretization and voting, for ease of implementation and noise resilience respectively. Experiments show that such extensions are feasible and effective.

2 Preliminaries and Notation

The data owner's private database $\mathcal{D}$ is denoted with transcript $\mathcal{D}(r_1, ..., r_n)$, where $r_i \in \mathcal{D}$ are the records in $\mathcal{D}$. We make no assumptions regarding the structure or type of data contained in $\mathcal{D}$, apart from the ability to map $\mathcal{D}$ to a m-dimensional Euclidean space $\mathbb{R}^m$. As such, we view each r_i as a *point* in Euclidean space, and use the terms *record* and *point* interchangeably.

Our starting point is pairwise distances between points in Euclidean space: As their name implies, distance preserving transformations (DPTs) preserve pairwise distances. Pairwise distances between elements $r_i, r_j \in \mathcal{D}$ can be computed using commonly used distance metrics, e.g., Minkowski (p-norm) distance, Euclidean distance, etc. [12] Without loss of generality, we assume that Euclidean distance is used.

Definition 1 (Euclidean distance)**.** *Let x and y be two data points in $\mathbb{R}^m$, with coordinates $x = (x^1, ..., x^m)$ and $y = (y^1, ..., y^m)$. We say that the Euclidean distance between x and y is: $\delta(x, y) = ||x - y|| = \sqrt{\sum_{i=1}^{m}(x^i - y^i)^2}$, where $||.||$ denotes the L^2-norm.*

For help in giving concrete examples, we introduce the notion of distance matrix (also known as the dissimilarity matrix [2]) that captures pairwise distances between points.

Definition 2 (Distance matrix)**.** *The distance matrix of a database $\mathcal{D}(r_1, .., r_n)$ is an $n \times n$, symmetric, real-valued matrix M such that $M_{i,j} = M_{j,i} = \delta(r_i, r_j)$.*

TABLE 1
Creating the distance matrix of a spatial database

ID	Coordinates
r_1	(34.0, 122.6)
r_2	(13.1, 57.8)
r_3	(2.5, 51.9)
r_4	(98.4, 193.2)

(a) Database $\mathcal{D}$ with four records

	r_1	r_2	r_3	r_4
r_1	0	68.1	77.4	95.6
r_2	68.1	0	12.1	160
r_3	77.4	12.1	0	170.8
r_4	95.6	160	170.8	0

(b) Distance matrix of $\mathcal{D}$

For example, let $\mathcal{D}$ be a geospatial database containing (X, Y) coordinates of 2D data points. A sample database $\mathcal{D}$ is shown in Table 1a. $\mathcal{D}$'s distance matrix is given in 1b. As an example, we compute one of the entries in the distance matrix: $M_{1,2} = \delta(r_1, r_2) = \sqrt{(34.0 - 13.1)^2 + (122.6 - 57.8)^2} = 68.1$.

Definition 3 (Distance Preserving Transformation)**.** *A function $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^d$ is a distance preserving transformation (DPT) if for all $x, y \in \mathbb{R}^m$, $\delta(x, y) = \delta(\mathcal{T}(x), \mathcal{T}(y))$.*

Let $\mathcal{D}$ be the data owner's private database. Instead of releasing $\mathcal{D}$, for privacy protection the data owner first perturbs $\mathcal{D}$ using a DPT $\mathcal{T}$ and then releases the perturbed data $\mathcal{D}' = (\mathcal{T}(r_1), ..., \mathcal{T}(r_n))$. By definition, $\mathcal{T}$ preserves pairwise distances between records, and thus the distance matrix M is constant before and after $\mathcal{T}$. We note that $\mathcal{T} : \mathbb{R}^m \to \mathbb{R}^d$ where we do not require $m = d$. That is, we allow $\mathcal{T}$ to change the dimensionality of the data. This increases the scope of transformations by covering the likes of principle component analysis, singular value decomposition and kernel functions.

In this paper we consider a broader type of transformations that we call *relation preserving transformations*. Such transformations allow pairwise distances (hence, the distance matrix) to change, but only in a way that the relative order of the distances is conserved. That is, assuming M is the distance matrix of the original data and M' is the distance matrix of the transformed data, if $M_{i,j}$ is greater than [less than] $M_{k,l}$, $M'_{i,j}$ must be greater than [less than] $M'_{k,l}$. Relation preserving transformations have the desirable property that similar data mining results can be obtained although exact pairwise distances between records are not revealed. For example, a record's k nearest neighbors do not change, therefore k-NN classification on transformed data would produce the same output as if it were run on the original data.

Definition 4 (Relation preserving transformation)**.** *A function $\mathcal{S} : \mathbb{R}^m \to \mathbb{R}^d$ is a relation preserving transformation (RPT) if for $x, y \in \mathbb{R}^m$, $z, t \in \mathbb{R}^d$ and for arithmetic comparison operators $op \in \{<, >, =\}$, $\delta(\mathcal{S}(x), \mathcal{S}(y))$ op $\delta(\mathcal{S}(z), \mathcal{S}(t))$ if and only if $\delta(x, y)$ op $\delta(z, t)$.*

Theorem 1. *Every DPT is relation preserving.*

Proof. Let $\mathcal{T} : \mathbb{R}^m \rightarrow \mathbb{R}^d$ be a DPT. Since $\mathcal{T}$ is distance preserving, for all $x, y, z, t \in \mathbb{R}^m$, $\delta(\mathcal{T}(x), \mathcal{T}(y)) = \delta(x, y)$ and $\delta(\mathcal{T}(z), \mathcal{T}(t)) = \delta(z, t)$. Therefore, trivially for any comparison operator *op*, $\delta(\mathcal{T}(x), \mathcal{T}(y))$ *op* $\delta(\mathcal{T}(z), \mathcal{T}(t))$ if and only if $\delta(x, y)$ *op* $\delta(z, t)$. □

The goal of this paper is to attack RPTs. We use Theorem 1 to show that DPTs are a subset of RPTs, therefore attacks on RPTs are applicable to DPTs (whereas the literature mostly focuses on attacks on DPTs). Notice that the converse of Theorem 1 is not true: A RPT is not necessarily distance preserving. For example, let $\mathcal{D}'$ in Table 2a be obtained via transforming $\mathcal{D}$ in Table 1a in a relation preserving manner. One can verify that the pairwise order of distances in Table 2b is the same as in Table 1, but none of the distances actually stay the same.

TABLE 2
A relation preserving transformation of $\mathcal{D}$

ID	Coordinates
r'_1	(32.7, 123.6)
r'_2	(14.5, 60.3)
r'_3	(8.8, 54.1)
r'_4	(100.0, 196.9)

(a) Transformed database $\mathcal{D}'$

	r'_1	r'_2	r'_3	r'_4
r'_1	0	65.9	73.5	99.5
r'_2	65.9	0	8.4	161.2
r'_3	73.5	8.4	0	169.4
r'_4	99.5	161.2	169.4	0

(b) Distance matrix of $\mathcal{D}'$

Definition 5 (Relation retrieval function)**.** *For a database $\mathcal{D}$ and arbitrary $r_A, r_B, r_C, r_D \in \mathcal{D}$, the relation retrieval function $\mathcal{R}$ is defined as:*

$$\mathcal{R}_{\mathcal{D}}((r_A, r_B), (r_C, r_D)) = \begin{cases} -1 & \text{if } \delta(r_A, r_B) < \delta(r_C, r_D) \\ 0 & \text{if } \delta(r_A, r_B) = \delta(r_C, r_D) \\ 1 & \text{if } \delta(r_A, r_B) > \delta(r_C, r_D) \end{cases}$$

$\mathcal{R}$ is essentially a function that captures the relations between records' distances in a dataset. From Definitions 3 and 4, we have $\mathcal{R}_{\mathcal{D}} = \mathcal{R}_{\mathcal{D}'}$, where $\mathcal{D}'$ is obtained from $\mathcal{D}$ using a DPT or RPT.

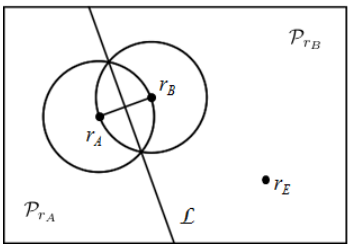
3 Attack Algorithm

We propose a novel strategy to attack RPTs. We assume that the following information is available to the attacker:

- **Outputs of the relation retrieval function,** $\mathcal{R}_{\mathcal{D}'}$. That is, the attacker knows the *relations* between distances in the transformed dataset. Clearly, the attacker can obtain this if he is given either $\mathcal{D}'$ or M'. Assuming that the attacker has $\mathcal{R}_{\mathcal{D}'}$ is a more relaxed assumption than assuming he has $\mathcal{D}'$ or M', but the latter is the prevalent assumption in related work (e.g., [2], [9]).
- **A set of known samples**. The attacker has a sample of records $r_i \in \mathcal{D}$. That is, the attacker knows where each r_i maps to in the original $\mathbb{R}^m$ space (prior to transformation).

Known-sample attacks are popular in the literature [5], [7], [9], [12]. There are multiple ways in which an attacker can obtain a set of known samples, e.g., the attacker may know that his and a few other friends' information is in the data, or may be able to inject a record into the data. Notice that our attack makes no assumptions regarding the underlying transformation function used or the transformed output. That is, we do not require the attacker to obtain input-output pairs or any output points from the transformation function $\mathcal{S}$.

In Section 3.1, we present our attack on 2D data so that we can easily illustrate it with examples. In Section 3.2, we formalize the attack and generalize it to arbitrary dimensions.



Distance matrix of $\mathcal{D}'$

	r'_A	r'_B	r'_E
r'_A	0	3	14
r'_B	3	0	12
r'_E	14	12	0

Fig. 1. Sample 2-dimensional database $\mathcal{D}$ with three records. Actual locations of records in $\mathbb{R}^2$ (on the left) and the distance matrix published after transformation (on the right).

3.1 Intuitive Explanation of the Attack

We introduce our attack using the example in Fig. 1. Suppose that database $\mathcal{D}$ is 2-dimensional (i.e., records are in $\mathbb{R}^2$), and let r_A, r_B in $\mathcal{D}$ be the known samples of the attacker. Say that the goal of the attacker is to find the position of r_E, i.e., locate r_E in $\mathbb{R}^2$ space. Let M' be the distance matrix that is published after a RPT $\mathcal{S}$ is applied to $\mathcal{D}$.

Observation 1. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = -1$, then in the original dataset r_E must be located outside the circle with center r_A and radius $\delta(r_A, r_B)$.*

Observation 2. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = 0$, then in the original dataset r_E must be located on the circle with center r_A and radius $\delta(r_A, r_B)$.*

Observation 3. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = 1$, then in the original dataset r_E must be located within the area enclosed by the circle with center r_A and radius $\delta(r_A, r_B)$.*

Given the known samples r_A, r_B, and $\mathcal{R}$, the attacker iteratively prunes the search space while searching for r_E. Observations 1, 2 and 3 demonstrate one way of pruning. The main idea is to compare the distance between the two known samples (r_A and r_B), and the distance between the target (r_E) and the known samples after the transformation is applied. The relation preservingness of the transformation allows the attacker to make inferences on the original dataset and prune out those portions in $\mathbb{R}^2$ that r_E cannot be located in. In all of the observations above, we compare $\delta(r_A, r_B)$ with $\delta(r_A, r_E)$, however $\delta(r_A, r_B)$ can be also compared with $\delta(r_B, r_E)$ which would result in circles centered at r_B with radius $\delta(r_A, r_B)$. The procedure can also be repeated for every pair of samples the attacker has (we considered only one pair (r_A, r_B) so far).

These observations are exemplified in Fig. 1. Since in the distance matrix we have $\delta(r'_A, r'_B) = 3$ is less than $\delta(r'_A, r'_E) = 14$, we have $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = -1$. The attacker that knows this information performs the following pruning: He draws the circle centered at r_A and radius equal to $\delta(r_A, r_B)$. Let $\delta(r_A, r_B) = 2$. Then, r_E cannot be within this circle, since *r_E is further away from r_A than r_B*. Next, since we have $\delta(r'_B, r'_E) = 12$, following the same reasoning, we draw the circle centered at r_B and radius equal to $\delta(r_A, r_B) = 2$. r_E must also be outside this circle, since *r_E is further away from r_B than r_A*.

We now present a second type of pruning. For the two known data samples r_A and r_B, let $\mathcal{L}$ denote the perpendicular bisector of the hypothetical line connecting r_A and r_B. (Given the locations of r_A and r_B, it is trivial to draw both the hypothetical line and its perpendicular bisector.) As seen in Fig. 1, $\mathcal{L}$ divides the search space into two portions. Let $\mathcal{P}_{r_A}$ denote the portion that contains r_A and $\mathcal{P}_{r_B}$ denote the portion that contains r_B.

Observation 4. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 1$, then r_E must be located in $\mathcal{P}_{r_B}$.*

Proof (Sketch). From the definition of $\mathcal{R}$, we have $\delta(\mathcal{S}(r_A), \mathcal{S}(r_E)) > \delta(\mathcal{S}(r_B), \mathcal{S}(r_E))$. Since $\mathcal{S}$ is relation preserving, this implies $\delta(r_A, r_E) > \delta(r_B, r_E)$. $\mathcal{L}$ is a line containing points that are equidistant to r_A and r_B. All points $X \in \mathcal{P}_{r_B}$ have the property $\delta(r_B, X) < \delta(r_A, X)$, whereas points Y on $\mathcal{L}$ satisfy $\delta(r_B, Y) = \delta(r_A, Y)$ and points $Z \in \mathcal{P}_{r_A}$ satisfy $\delta(r_A, Z) < \delta(r_B, Z)$. Hence, r_E is in $\mathcal{P}_{r_B}$. □

Observation 5. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 0$ then r_E must be located on $\mathcal{L}$.*

Observation 6. *If $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = -1$ then r_E must be located in $\mathcal{P}_{r_A}$.*

The second type of observations we make (Obs. 4, 5 and 6) examine the distance between the target r_E and the two known samples (i.e., $\delta(r_A, r_E)$ and $\delta(r_B, r_E)$). Again, this process can be repeated for every pair of known samples.

We exemplify using Fig. 1. We have $\delta(r'_A, r'_E) = 14$ and $\delta(r'_B, r'_E) = 12$. Thus, $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 1$. This enables the attacker to infer that *r_E is closer to r_B than it is to r_A*. Then, he draws the perpendicular bisector $\mathcal{L}$ and locates $r_E \in \mathcal{P}_{r_B}$.

At this point we would like to underline two characteristics of our attack: (1) The pruning process is fully deterministic, i.e., the attacker is 100% confident that pruned areas may not contain the target data point. (2) We are placing constraints on where r_E can/cannot be located in the original dataset, not the transformed dataset. The attacker's goal is to locate r_E in the original data space, not in the transformed space.

3.2 Attack Formalization

In this section we formalize our attack and generalize it to n-dimensional space $\mathbb{R}^n$, where $n \geq 2$.

Definition 6 (Hypersphere)**.** *In n-dimensional Euclidean space, a hypersphere $S_{C,r}$ is defined by a fixed center point $C \in \mathbb{R}^n$ and a radius r, and denotes the set of points in the n-dimensional space that are at distance r from C. That is, each point $X(X_1, X_2, ..., X_n)$ on $S_{C,r}$ satisfies: $r^2 = \sum_{i=1}^{n} (X_i - C_i)^2$.*

Definition 7 (Hyperball)**.** *In n-dimensional Euclidean space, given a hypersphere $S_{C,r}$, the hyperball $B_{C,r}$ denotes the space enclosed by $S_{C,r}$. $B_{C,r}$ is said to be closed if it includes $S_{C,r}$ and open otherwise.*

For example, circles are hyperspheres in 2-dimensional space. A circle with center $C(3, 4)$ and radius 7 would be characterized by the equation: $(x-3)^2 + (y-4)^2 = 49$. Then, the open hyperball $B_{(3,4),7}$ specifies the area enclosed by this circle, excluding those points that are on the circle. This is given by the equation: $(x-3)^2 + (y-4)^2 < 49$.

Definition 8 (Equidistant hyperplane)**.** *In n-dimensional Euclidean space, the locus of points equidistant from two points $A, B \in \mathbb{R}^n$ is a hyperplane H_{AB} that contains all points P that satisfy the equality $||P - A|| = ||P - B||$.*

For example, let $A, B \in \mathbb{R}^3$ have the following coordinates: $A(-2, 1, 4), B(0, 6, 3)$. We say that P has the coordinates $P(x, y, z)$ and solve $||P - A|| = ||P - B||$. According to Euclidean distance, this builds the equation: $\sqrt{(x+2)^2 + (y-1)^2 + (z-4)^2} = \sqrt{(x-0)^2 + (y-6)^2 + (z-3)^2}$. After simple arithmetic, we obtain: $4x + 10y - 2z = 24$, which is the equation of the equidistant hyperplane H_{AB}.

Definition 9 (Half-space)**.** *A half-space is either of the two parts into which a hyperplane divides the n-dimensional Euclidean space. A half-space is said to be closed if it includes the hyperplane, and open otherwise.*

Half-spaces are often specified using linear inequalities derived from the hyperplane that seperates them. For the previous example, the two open half-spaces of $\mathbb{R}^3$ are given by the equations:

$$\mathcal{P}_A : 4x + 10y - 2z < 24$$
$$\mathcal{P}_B : 4x + 10y - 2z > 24$$

It is clear to see that since H_{AB} is the hyperplane that is equidistant to A and B, one of the half-spaces will contain point A whereas the other will contain B. We call these half-spaces $\mathcal{P}_A$ and $\mathcal{P}_B$ respectively.

We present our attack strategy in Algorithm 1. We have the following inputs: The universe U is a collection of all possible points that may exist in the database. The universe often has boundaries that are dictated by the semantics of the underlying database, e.g., the boundaries of a dimension *age* could be 0 and 110. Or, if the database contains the locations or spatial information regarding people living in a particular city, the universe is bounded by the borders of that city. $\mathcal{R}_{\mathcal{D}'}$ captures the distance relations after a RPT. The attacker has a set of legitimate known samples (i.e., all samples are within the universe U). We describe the attack assuming the attacker would like to compromise the location of one target record r_E, but the attack can be run on an arbitrary record. We specify the identifier of the target record, E, as one of our inputs.

Initially, we say that r_E can be anywhere in the universe, by setting the search space variable (denoted by s) to U. For every pair of known samples, we iteratively prune the search space several times using the observations made in the previous section. Here, pruning refers to deleting certain geometric objects, or areas that do not intersect with a given geometric object, from the search space. To make the algorithm easier to follow, we give the specific observations that lead to each of the pruning operations. On lines 4-5 we apply Observation 1, on lines 6-7 we apply Observation 3, and on lines 8-9 we apply Observation 2. These three steps are based on the distances between (r_A, r_B) and (r_A, r_E). We then apply the same approach to the distances between (r_A, r_B) and (r_B, r_E) to obtain the three steps between lines 10-16: On lines 11-12 we apply Observation 1, on lines 13-14 we apply Observation 3, and on lines 15-16 we apply Observation 2. Afterwards, between lines 17-23, we apply Observations 4, 5 and 6. Specifically, on lines 18-19 we apply Observation 4, on lines 20-21 we apply Observation 6, and on lines 22-23 we apply Observation 5. The final output of the attack is the region of the universe that has not been pruned, i.e., the area where r_E must be located in.

3.3 Implementation Considerations

Discretization. As can be seen in Algorithm 1, our attack involves many union, intersection and difference operations. These are non-trivial to implement in continuous n-dimensional space. For

Algorithm 1 Attack for locating a target record using distance relations and known samples

Input: U: data space and its boundaries,
$\mathcal{R}_{\mathcal{D}'}$: distance relations of the transformed data,
$K = \{r_1, .., r_n | r_i \in U\}$: set of known samples,
E: an identifier to denote the target record r_E
Output: $U' \subseteq U$: portion of the universe where the target record is located

1: $s \leftarrow U$
2: **for** each pair $(r_A, r_B) \in K$ **do**
3: Build the hypersphere $S_{r_A,\delta(r_A,r_B)}$ and open hyperball $B_{r_A,\delta(r_A,r_B)}$
4: **if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = -1$ **then**
5: $s \leftarrow s - (B_{r_A,\delta(r_A,r_B)} \cup S_{r_A,\delta(r_A,r_B)})$
6: **else if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_A, r_E)) = 1$ **then**
7: $s \leftarrow s \cap B_{r_A,\delta(r_A,r_B)}$
8: **else**
9: $s \leftarrow s \cap S_{r_A,\delta(r_A,r_B)}$
10: Build the hypersphere $S_{r_B,\delta(r_A,r_B)}$ and open hyperball $B_{r_B,\delta(r_A,r_B)}$
11: **if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_B, r_E)) = -1$ **then**
12: $s \leftarrow s - (B_{r_B,\delta(r_A,r_B)} \cup S_{r_B,\delta(r_A,r_B)})$
13: **else if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_B), (r_B, r_E)) = 1$ **then**
14: $s \leftarrow s \cap B_{r_B,\delta(r_A,r_B)}$
15: **else**
16: $s \leftarrow s \cap S_{r_B,\delta(r_A,r_B)}$
17: Build the equidistant hyperplane $H_{r_A r_B}$ and resulting open half-spaces $\mathcal{P}_{r_A}$, $\mathcal{P}_{r_B}$
18: **if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = 1$ **then**
19: $s \leftarrow s \cap \mathcal{P}_{r_B}$
20: **else if** $\mathcal{R}_{\mathcal{D}'}((r_A, r_E), (r_B, r_E)) = -1$ **then**
21: $s \leftarrow s \cap \mathcal{P}_{r_A}$
22: **else**
23: $s \leftarrow s \cap H_{r_A r_B}$
24: **return** s

the sake of reproducibility, we comment on the specifics of our implementation.

We implement the attack by discretizing the search space: We assume that the universe U is made up of uniform n-dimensional cells. Smaller the cells are, higher the total number of cells and finer the granularity of the attack will be. However, due to the increased number of cells, execution time will also be higher. We take a defensive approach when we prune cells, that is, in each pruning decision we prune only those cells that can be *completely* pruned off the search space. For example, consider Fig. 2. When we prune the half-space $\mathcal{P}_{r_A}$, we only prune those cells that are completely contained by $\mathcal{P}_{r_A}$. For all borderline cells (e.g., those that lie on $\mathcal{L}$) we keep them instead of pruning them. As such, we guarantee that we never over-prune, e.g., if in Fig. 2 we prune *cell Y* then we would have over-pruned by removing the portion that is to the right of $\mathcal{L}$, which includes area that r_E could actually be located in. This would violate the correctness of our attack. On the other hand, by not pruning *cell Y* we also keep the portion in *cell Y* that is to the left of $\mathcal{L}$, which we are certain that r_E is not located in. We ideally would like to prune the latter portions off, but since our cells were too coarse (i.e., too big) in this case, we could not do so. On the other hand, we safely prune *cell X* since the whole cell lies within $\mathcal{P}_{r_A}$.

Complexity Analysis. We analyze the time complexity of our attack given the implementation details above. Let the universe U be n-dimensional, and for the sake of simplicity let us assume that each dimension has the same length (i.e., difference between the endpoints of its boundaries) of $\Omega(U)$. Further, let us divide the search space into uniform cells with side lengths of c. Then, we have $\left(\frac{\Omega(U)}{c}\right)^n$ cells in total.

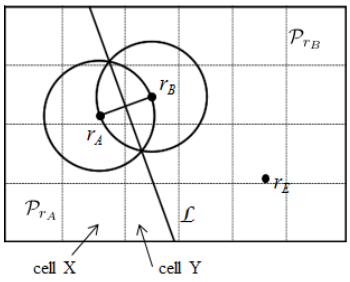


Fig. 2. Discretization of the universe using uniform 2-dimensional cells.

Algorithm 1 executes its main loop (between lines 2-23) a total of $\binom{|K|}{2}$ times. Within the loop, it builds two hyperspheres and one hyperplane. Given the equations of these geometrical structures, constructing them is trivial and can be done in constant time. However, for each structure, it performs exactly one space pruning operation, which depends on computing the intersection of each cell and the constructed structures. We incorporate a factor $\mathcal{I}$ for computing intersections, which depends heavily on the programming language and geometry libraries used. Then, the overall time complexity of our attack is $O\left(\binom{|K|}{2} \cdot \left(\frac{\Omega(U)}{c}\right)^n \cdot \mathcal{I}\right)$.

As implied above, the attack is exponential in n (number of dimensions) and $|K|$ (number of known samples). Although this may seem prohibitive, we note that: (i) The attack is typically not executed in real-time, and therefore there are no strict efficiency requirements. For example, if the sensitive information being attacked is a victim's home location, since this does not change frequently, it is not a burden to wait for a few hours to obtain the attack outcome. (ii) Generally both n and $|K|$ are low, e.g., in our experimental setting we use 2D location data from location-based social networks with few known samples, so that $n = 2$ and $|K| \in [2, 10]$. With $|K| = 4$, our implementation (in Java) can execute the attack in 8 seconds, using a laptop with a single 2.2 GHz processor and 8 GB memory.

Noise Resilience. One of the prominent techniques in data privacy is *additive perturbation*, which adds noise to the published information. Thus, it is interesting to make our attack resilient to the addition of noise. Note that noise addition will likely destroy the relation preservingness of a transformation, and the correctness of our attack can no longer be guaranteed. That is, given data space U the attack may output that r_E resides in space U', but in fact r_E resides in $U - U'$.

Algorithm 1 is not resilient to noise, since it prunes a region of the search space immediately when one pair decides that the region should be pruned. For instance, let (r_C, r_D) be a pair that decides to prune cell X when searching for target record r_E. Algorithm 1 would immediately prune X from the search space variable s and proceed. This is not problematic in the noise-free case. However, in the noisy case, $\mathcal{R}_{\mathcal{D}'}$ may be imperfect due to the added noise. Thus, (r_C, r_D)'s decision to prune X could be wrong.

To account for these cases, we implemented a *voting mechanism*. For each cell, we keep track of the pairs of records that have *voted* in favor of pruning that cell. If, at any point, the number of votes for pruning cell X exceeds an input voting threshold,

we prune X off the search space. That is, let $t \in [0,1]$ denote the voting threshold and $|K|$ be the number of known samples. If $t \times |K|$ samples vote in favor of pruning X, then X will be pruned. We use the voting mechanism only when the data is noisy (i.e., the transformation is not guaranteed to be relation preserving) or the answers to $\mathcal{R}_{\mathcal{D}'}$ are noisy (e.g., untruthful).

Further Improvements. We discuss potential improvements that increase the practicality of our attack under two settings: *limited time* and *limited access*.

In the limited time setting, we assume that the attack must be executed in a short amount of time, e.g., the attacker wants to know where the victim *currently* is, in a dynamic city environment. The exponential complexity of the attack might be a problem, therefore we propose two improvements to overcome this problem: pre-computation with known samples, and multi-granularity grids. For pre-computation, we assume that the attacker's known samples stay constant over time, or follow a probability distribution known by the attacker. For example, if the known samples are the attacker's friends in a social network, then the attacker would know their probable locations (e.g., work, home, frequent visits). Given these for each pairs of known samples, the attacker can pre-compute $S_{r_A,\delta(r_A,r_B)}$, $S_{r_B,\delta(r_A,r_B)}$ and $H_{r_A r_B}$, since these do not rely on the victim's record r_E. Then, the attacker determines which cells would be pruned depending on the answers of $\mathcal{R}$, and hardcodes them into the attack to improve efficiency. The second improvement, multi-granularity grids, proposes to replace the single-level uniform grid with hierarchically organized multiple-layer grids (e.g., 8x8 $\rightarrow$ 4x4 $\rightarrow$ 2x2). Then, if a cell in the topmost grid with largest cells (i.e., 2x2) can be pruned, we immediately infer all cells in the finer-granularity grids (i.e., 8x8) that are covered by the larger cell can also be pruned. The use of more sophisticated and data-dependent indexing structures, e.g., quad-trees and kd-trees, are left to future work.

In the limited access setting, we assume that the data is outsourced to an external service, and the attacker can issue a bounded number of relation retrieval queries $\mathcal{R}$ due to their monetary cost or privacy countermeasures implemented by the external service. Then, we need to study methods to maximize the effectiveness of our attack under this constraint. For this purpose, we propose strategically selecting the pairs of known samples (on line 2 of Alg. 1) that lead to highest amount of expected pruning. Specifically, we suggest that the attacker chooses a balanced combination of: (i) Nearby known samples, i.e., (r_A, r_B) such that $\delta(r_A, r_B)$ is as small as possible. In this case, the circles drawn in Fig. 1 are small. Then, if Observation 3 holds, the attacker reduces the search space into a small circle, which increases his precision. (ii) Distant known samples, i.e., (r_C, r_D) such that $\delta(r_C, r_D)$ is as large as possible. In this case, the circles drawn in Fig. 1 are large. Then, if Observation 1 holds, the attacker can prune large circles off the search space. A combination of these two extremes should give the attacker an increased amount of expected pruning.

4 Experiments

4.1 Experiment Setup

Consider the following practical application of our attack: The data owner runs a mobile service and collects private location check-ins of users. This data is shared with third parties after a RPT. Since the attacker and a few close friends of his are users of the mobile service, the attacker knows their check-in locations and this constitutes his set of known samples. Then, the goal of the attacker is to infer the locations of remaining users (whose locations he cannot directly observe, since, e.g., they are not in his social circle). Motivated by this scenario, we use two 2D real-life location datasets in our experiments, which are as follows:

Gowalla dataset [13]. Gowalla was a location-based social networking website where users shared their locations by checking-in. A total of 6,442,890 check-ins over the period of February 2009 and October 2010 were collected and made available[1]. This dataset was recently used in other privacy related works as well. From the Gowalla data, we extracted those check-ins made in New York.

Istanbul dataset. We collected location data from 60,000 vehicles in Istanbul, Turkey using a vehicle tracking system. From this data, we extracted the last known locations of 200 randomly chosen vehicles and formed our experimental dataset. The data consists of the vehicle ids, latitude and longitude coordinates.

We use our grid-based implementation explained in Section 3.3. We pick the grid cell size such that the area of each cell is $0.01km^2$. We believe that locating an individual within such a small area is reasonable and challenging considering the entire search space is a big city. Given this cell size, we have a total of approximately 1.1mil cells in Istanbul and 240k cells in Gowalla.

We run experiments in two scenarios: *noisy* and *noise-free*. In the *noise-free* scenario, a DPT is applied to the data (which, by Def. 1, is also a RPT) and the attacker observes the true answers to $\mathcal{R}$. In the *noisy* scenario, we assume that $\mathcal{R}$ is answered by a third party (external) service. For privacy reasons, the service randomly returns false answers to $p\%$ of the queries, whereas the remaining $(100-p)\%$ is answered truthfully.

We repeat each experiment 100 times by changing the set of known samples and the target, and report the average results.

4.2 Evaluation Metrics

Following the inputs and outputs of Algorithm 1, we use the following transcript to denote the attack: $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}'}, K, E) = U'$, where $\mathcal{A}$ denotes the attack, $(U, \mathcal{R}_{\mathcal{D}'}, K, E)$ denote the four input parameters as described in Algorithm 1, and U' is the output, i.e., the portion of the search space that the attack claims the target record r_E is located in. We quantify accuracy as follows:

$$\text{Accuracy} = Pr(r_E \in U' | \mathcal{A}(U, \mathcal{R}_{\mathcal{D}'}, K, E) = U')$$

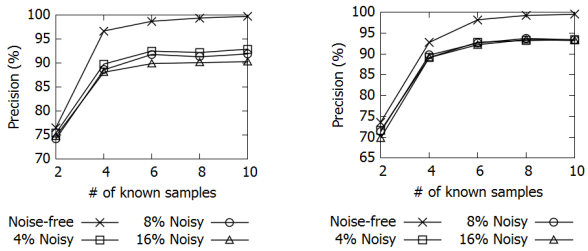
The probability boils down to the empirical ratio of trials where the attack was right in predicting that r_E was located in portion U' divided by the total number of trials. We underline that the accuracy of our attack is always 100% when there is no added noise, i.e., the data transformation is relation preserving.

Our second metric is *precision*. An attack that simply outputs $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}'}, K, E) = U$ without doing anything achieves 100% accuracy, but it cannot be considered successful since it is very imprecise. The success of the attack lies in its ability to identify a *small* portion $U' \subseteq U$ that r_E is located in. This is captured by the *precision* metric we define below. Let $vol(\cdot)$ denote the volume of a given n-dimensional region. Given $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}'}, K, E) = U'$:

$$\text{Precision} = \frac{vol(U - U')}{vol(U)}$$

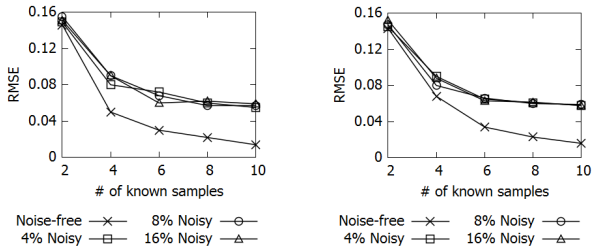
In a uniform-cell based implementation, precision can be calculated simply by dividing the number of pruned cells (i.e., those in $U - U'$) by the total number of cells in the universe U.

1. http://snap.stanford.edu/data/loc-gowalla.html



(a) Gowalla dataset (b) Istanbul dataset

Fig. 3. Precision in noise-free and noisy scenarios



(a) Gowalla dataset (b) Istanbul dataset

Fig. 4. RMSE in noise-free and noisy scenarios

Our third metric is *RMSE*, the root mean square error of estimation. Since the output of the attack is U', the attacker estimates that the target record is randomly assigned to a point in a cell in U'. To measure the error in this estimation, given the cells $C_i \in U'$ and the maximum distance allowed by the boundaries of the data space $\rho(U)$, we calculate:

$$\text{RMSE} = \sqrt{\frac{\sum_{C_i \in U'} \left(\frac{\delta(\mu(C_i), r_E)}{\rho(U)}\right)^2}{\text{\# of cells in } U'}}$$

where $\mu(C_i)$ denotes the center of cell C_i. Note that the actual distance $\delta(\mu(C_i), r_E)$ is normalized by $\rho(U)$ to ensure we do not face granularity errors and results remain comparable across multiple datasets.

4.3 Results and Discussion

The most interesting aspect of our attack is how precision changes with respect to the number of known samples. With this, we can directly infer how many samples would be needed to locate a target record. We run this experiment in the noise-free scenario, and the noisy scenario with varying levels of perturbation ($p\%$). We graph the results in Fig. 3. In the noise-free scenario, the attack achieves 96% precision with 4 or more known samples. Even with 2 samples, the attack achieves 69% and 76% precision on the Istanbul and Gowalla datasets respectively. Considering that it is easy to obtain a sample of 2-4 records (e.g., check-ins of the attacker himself and 1-3 acquaintances) we can deduce that our attack is very feasible in practice.

Using Fig. 3, we also compare the precision between noisy and noise-free scenarios. For the noisy scenario, (i) we only consider the precision of those records that are correctly located, i.e., $r_E \in U'$ given that $\mathcal{A}(U, \mathcal{R}_{\mathcal{D}'}, K, E) = U'$, and (ii) we choose an average voting threshold of 0.6 (the choice of this value will be justified later when we discuss Fig. 6). We make several observations: (i) Precision is higher in the noise-free scenario than in the noisy scenarios. However, their difference is only 10%, which implies that the attack is resilient against noise. (ii) When we increase the amount of noise (i.e., $p\%$), precision drops, but only slightly. (iii) More known samples yields higher precision. Interestingly, in the noisy scenarios, it appears that precision is "capped" at some level (93-94$\%$) and cannot increase further. We observed that this is caused by our voting mechanism: A voting threshold of 0.6 means that the majority of the known samples must agree to prune a cell before that cell can be pruned. This causes some borderline cells which are easily pruned in the noise-free scenario to remain unpruned in the noisy scenario.

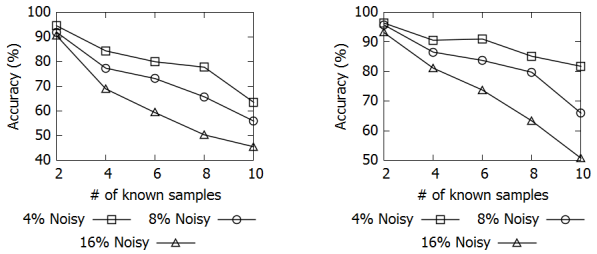
In Fig. 4, we repeat the same experimental setting but report results using the RMSE metric. As expected, higher precision yields lower RMSE. RMSE values are higher in the noisy scenarios than the noise-free scenario, which can again be explained using the selective pruning of our voting mechanism which results in larger unpruned regions. However, even though there is a clear relationship between the percentage of noise and precision (see Fig. 3a), there is no such relationship between percentage of noise and RMSE, i.e., they do not seem correlated. This can be explained in conjunction with the steady decrease in RMSE with more known samples. We observed that pruned/unpruned regions and cells are often adjacent, e.g., in Fig. 2, we do not end up with the bottom right corner and upper left corner simultaneously unpruned. Therefore the unpruned regions (the sole factor affecting RMSE) are near the target record. If the opposite were true, we would have seen clear increase in RMSE with higher noise.

We comment on the variance of the obtained results by building 95% confidence intervals. We observe that variance drops significantly when we have more known samples. For example, with 2 known samples, confidence intervals of precision and RMSE are $74\% \pm 5\%$ and 0.143 ± 0.02 respectively. However, with 6 known samples, the confidence intervals are $96\% \pm 1.7\%$ and 0.034 ± 0.007 respectively. This is explained as follows: When we have few known samples, since these samples are chosen randomly, their locations and relations with the target record have a higher impact on the amount of pruning. In some cases, we can prune as high as 95% of the universe, but in others, we can prune only 10%. Thus, variance is high. On the other hand, when we have 6 known samples, there is almost always some pairs that lead to effective pruning. As a result, the lower bound on the amount of pruning is high, e.g., we achieve at least 90% precision in every experiment. Thus, variance is low.

Next, in Fig. 5, we show the accuracy of our attack with respect to the number of known samples. Interestingly, accuracy drops as we have more samples. This is because of noise. Alg. 1 prunes for every pair of samples, and more samples imply more pruning. However, at the same time, more pruning increases the probability of erroneous pruning (i.e., pruning space that actually contains r_E) due to the existence of noise. This gives a clear trade-off between precision and accuracy: If we are more aggressive by pruning many regions, we increase our precision. However, at the same time, it becomes more likely that the region containing the target record will also be pruned if we are too aggressive. This would decrease our accuracy.

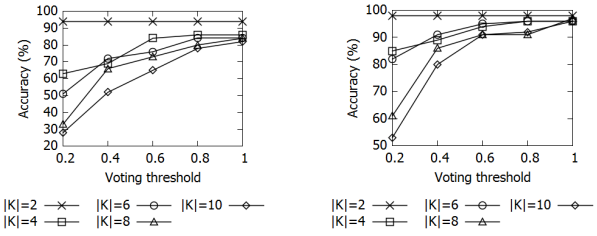
Even though we do not achieve 100% accuracy in case of noise, we observe that for those attacks falsely prunning the target point, the corresponding RMSE values are similar to those of the successful attacks. This means unpruned regions stand in close proximity to the target point. We stress that even an unsuccessful attack can disclose the whereabouts of a target.

Taking into account Fig.s 3, 4, 5, we describe some guidelines



(a) Gowalla dataset (b) Istanbul dataset

Fig. 5. Accuracy in the noisy scenario



(a) Gowalla dataset (b) Istanbul dataset

Fig. 6. Effects of the voting threshold on accuracy

for the attacker. The goal of the attacker should be to first remain accurate in his prediction, and second to make his prediction very precise. For example, if the attacker very precisely locates his target within 0.01% of the universe but the target is not actually located there, this is potentially a bigger problem than having a somewhat poorer precision, e.g., 4%, but remaining accurate in the prediction. Therefore we argue that accuracy should not be sacrificed in favor of precision.

One factor that affects accuracy in the noisy scenario is the voting threshold. We fix the noise level to 8% and perform experiments with varying number of known samples and thresholds, as illustrated in Fig. 6. We make the following observations: First, as the voting threshold increases, accuracy increases. This is because a higher voting threshold implies more known samples need to agree in order to prune a region, and therefore a single noisy answer is less likely to cause erroneous pruning. Second, as we have more samples, we should have higher voting thresholds in order to remain accurate. This is in parallel with the results and discussion of Fig. 5 - more samples usually cause accuracy to drop, and the voting threshold should be increased to compensate. Finally, we observe that for both datasets, voting thresholds of at least 0.6 or 0.8 are needed to remain roughly 80% accurate.

5 Conclusion

In this paper, we described and evaluated an attack on a generalized view of distance preserving transformations: relation preserving transformations. Our attack can reach high levels of accuracy and precision with only few known samples. For example, we can locate a target record with over 94% precision using only 4 known samples. In addition, we showed that our voting mechanism can effectively combat noise.

We believe that incorporating additional background knowledge such as map information (roads, points of interest, ...) or distribution of data points could enable the attacker further prune regions, thus achieve more precise attacks. Distribution of points could also be used to better evaluate the risk of identification.

Even though formulating such attacks is interesting, one should also study how we can defend against them. Introducing noise to disturb distance relations decreases attack accuracy, but this comes at the cost of utility for distance sensitive data mining operations. Due to exponential time complexity of the attack, one could also consider publishing data of higher dimensionality. However, possible attack improvements such as precomputation of samples and the use of multi-granularity grids still stand as a risk. How effective such improvements would be in practice is an open question. In future work, we plan to study the effectiveness of possible solutions using privacy techniques such as multiplicative perturbation [6], geometric perturbation [10] and aggregation-based methods [14]. We will also study the potential implications of our attack on order-preserving encryption [15].

References

[1] K. Chen and L. Liu, "Geometric data perturbation for privacy preserving outsourced data mining," *Knowledge and Information Systems*, vol. 29, no. 3, pp. 657–695, 2011.

[2] S. R. Oliveira and O. R. Zaïane, "Achieving privacy preservation when sharing data for clustering," in *Workshop on Secure Data Management*. Springer, 2004, pp. 67–82.

[3] K. Chen and L. Liu, "Privacy preserving data classification with rotation perturbation," in *IEEE International Conference on Data Mining (ICDM), 2005*. IEEE, 2005.

[4] J.-W. Huang, J.-W. Su, and M.-S. Chen, "Fisip: A distance and correlation preserving transformation for privacy preserving data mining," in *2011 International Conference on Technologies and Applications of Artificial Intelligence*. IEEE, 2011, pp. 101–106.

[5] C. R. Giannella, K. Liu, and H. Kargupta, "Breaching euclidean distance-preserving data perturbation using few known inputs," *Data & Knowledge Engineering*, vol. 83, pp. 93–110, 2013.

[6] K. Liu, C. Giannella, and H. Kargupta, "A survey of attack techniques on privacy-preserving data perturbation methods," in *Privacy-Preserving Data Mining*. Springer, 2008, pp. 359–381.

[7] K. Liu, C. R. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, 2006, pp. 297–308.

[8] S. Guo and X. Wu, "Deriving private information from arbitrarily projected data," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2007, pp. 84–95.

[9] E. O. Turgay, T. B. Pedersen, Y. Saygın, E. Savaş, and A. Levi, "Disclosure risks of distance preserving data transformations," in *International Conference on Scientific and Statistical Database Management*. Springer, 2008, pp. 79–94.

[10] K. Chen, G. Sun, and L. Liu, "Towards attack-resilient geometric data perturbation," in *Proceedings of the 2007 SIAM International Conference on Data Mining*. SIAM, 2007, pp. 78–89.

[11] B. D. Okkalioglu, M. Okkalioglu, M. Koc, and H. Polat, "A survey: deriving private information from perturbed data," *Artificial Intelligence Review*, vol. 44, no. 4, pp. 547–569, 2015.

[12] E. Kaplan, T. B. Pedersen, E. Savaş, and Y. Saygın, "Discovering private trajectories using background information," *Data & Knowledge Engineering*, vol. 69, no. 7, pp. 723–736, 2010.

[13] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 1082–1090.

[14] C. C. Aggarwal and P. S. Yu, "A condensation approach to privacy preserving data mining," in *EDBT*, vol. 4. Springer, 2004, pp. 183–199.

[15] A. Boldyreva, N. Chenette, Y. Lee, and A. Oneill, "Order-preserving symmetric encryption," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2009, pp. 224–241.