Data Inference from Encrypted Databases: A Multi-dimensional Order-Preserving Matching Approach

Yanjun Pan
University of Arizona
Tucson, AZ, USA
yanjunpan@email.arizona.edu

Alon Efrat
University of Arizona
Tucson, AZ, USA
alon@cs.arizona.edu

Ming Li
University of Arizona
Tucson, AZ, USA
lim@email.arizona.edu

Boyang Wang
University of Cincinnati
Cincinnati, OH, USA
boyang.wang@uc.edu

Hanyu Quan
Huaqiao University
Xiamen, China
quanhanyu@hqu.edu.cn

Joseph Mitchell
Stony Brook University
Stony Brook, NY, USA
joseph.mitchell@stonybrook.edu

Jie Gao
Stony Brook University
Stony Brook, NY, USA
jgao@cs.sunysb.edu

Esther Arkin
Stony Brook University
Stony Brook, NY, USA
estie@ams.stonybrook.edu

ABSTRACT

Due to increasing concerns of data privacy, databases are being encrypted before they are stored on an untrusted server. To enable search operations on the encrypted data, searchable encryption techniques have been proposed. Representative schemes use order-preserving encryption (OPE) for supporting efficient Boolean queries on encrypted databases. Yet, recent works showed the possibility of inferring plaintext data from OPE-encrypted databases, merely using the order-preserving constraints, or combined with an auxiliary plaintext dataset with similar frequency distribution. So far, the effectiveness of such attacks is limited to single-dimensional dense data (most values from the domain are encrypted), but it remains challenging to achieve it on high-dimensional datasets (e.g., spatial data) which are often sparse in nature. In this paper, for the first time, we study data inference attacks on multi-dimensional encrypted databases (with 2-D as a special case). We formulate it as a 2-D order-preserving matching problem and explore both unweighted and weighted cases, where the former maximizes the number of points matched using only order information and the latter further considers points with similar frequencies. We prove that the problem is NP-hard, and then propose a greedy algorithm, along with a polynomial-time algorithm with approximation guarantees. Experimental results on synthetic and real-world datasets show that the data recovery rate is significantly enhanced compared with the previous 1-D matching algorithm.

KEYWORDS

Data inference, encrypted database, order-preserving encryption, multi-dimensional matching

ACM Reference Format:
Yanjun Pan, Alon Efrat, Ming Li, Boyang Wang, Hanyu Quan, Joseph Mitchell, Jie Gao, and Esther Arkin. 2020. Data Inference from Encrypted Databases: A Multi-dimensional Order-Preserving Matching Approach. In *Mobihoc 2020: The Twenty-First ACM International Symposium on Mobile Ad Hoc Networking and Computing, October 11–14, 2020, Online.* ACM, New York, NY, USA, 10 pages. https://doi.org/xx.xxxx/xxxxxxx.xxxxxxx

1 INTRODUCTION

Data outsourcing has become popular in recent years. Small businesses or individual users choose to delegate their data storage to public cloud servers (such as Amazon EC2 or Google Cloud) to save operational costs. Meanwhile, data breaches happen at an increasing rate, which compromise users' privacy. For instance, the Yahoo! data breaches reported in 2016 affected 3 billion user accounts [32]. This is exacerbated by recent scandals of data misuse (such as the Facebook-Cambridge Analytica case [31]), which increases the level of distrust from users. To address this issue, end-to-end encryption is commonly adopted to encrypt the data before it is uploaded and stored on an untrusted server. In order to enable efficient utilization over encrypted data (such as answering queries), many cryptographic techniques called searchable encryption (SE) [4, 12, 29] have been proposed. The main challenge for SE is to simultaneously provide flexible search functionality, high security assurance, and efficiency. Among existing SE schemes, Order-Preserving Encryption (OPE) [7, 8, 22, 27] has gained wide attention in the literature due to its high efficiency and functionality. In particular, OPE uses symmetric key cryptography and preserves the numeric order of plaintext after encryption, which supports most Boolean queries such as range query. Well-known systems for encrypted database search using OPE include: CryptDB [28], Google Encrypted Bigquery Client [17], and Microsoft Always Encrypted Database [24].

Many early OPE schemes, unfortunately, were shown to leak more information beyond what is necessary (i.e., the order between plaintexts). Therefore, schemes that satisfy ideal security guarantees (that only the order is leaked) have been proposed [22, 27]. However, recent research [20, 25] showed that it is possible to infer/recover a significant portion of plaintexts from their OPE ciphertext, using only the ciphertext order relationships, as well as some auxiliary dataset with data frequencies similar to a target dataset. For example, Naveed et al. [25] attacked an encrypted medical database where users' age column is encrypted using OPE. Later, the attack was improved by Grubb et al. [20], with an additional restriction of non-crossing property in the matching algorithm.

We note that, to date, all the successful inference attacks against OPE are limited to one-dimensional data [20, 25]. That is, even though a database may have multiple numeric columns/dimensions, where each of them being encrypted by OPE, each of these columns are treated separately when they are matched with plaintext values. This works well for *dense* data, i.e., where most of the values from the whole data domain have corresponding ciphertexts present in the database, such as age [20]. Intuitively, the denser the data is, the more effective the attack is, because more constraints imposed by the ciphertext order reduces the uncertainty of their corresponding plaintext values. However, for multi-dimensional databases, applying such 1-D matching algorithms on each dimension separately can yield results far from optimal, since it neglects that for each pair of data tuples the order-preserving constraints on all the dimensions must be held jointly, leading to a much larger search space than the actual one and therefore more ambiguity in matching. In addition, for higher dimensional data (such as spatial/location data), the data tuple tends to be increasingly sparsely distributed in the domain, which invalidates the one-dimensional matching approach (unless the ciphertext and known plaintext datasets are highly similar with each other). Therefore, we wonder whether it is still feasible to recover OPE-encrypted data tuples for multi-dimensional, sparse databases? This turns out to be a very challenging problem.

In this paper, we study data inference attacks against multi-dimensional encrypted databases by jointly considering all the dimensions and leveraging only the ciphertext tuples' order and frequency information, with the help of an auxiliary plaintext dataset with similar frequencies (the same assumption is adopted by many previous works). We formulate the *order-preserving matching problem* first in 2D but later extend it to 3D and higher dimensions. In the unweighted case, given an OPE-encrypted database and an auxiliary plaintext dataset, each containing a set of points in 2D, we maximize the number of points in a matching from the ciphertext to the plaintext, where order-preserving property must be simultaneously satisfied in both dimensions. Such a matching is called a non-conflicting matching in which the x/y projection of one edge in the matching cannot contain the projection of another edge in the matching. In general we also consider point frequency (the number of records with the same value), points matched with a smaller frequency difference are given higher weights and we maximize the total weights of the matching.

We show that our problem can also be formulated as an integer programming problem (ILP), and prove its NP-hardness by reducing it to sub-permutation pattern matching problem. Then we propose a greedy algorithm, along with an approximation algorithm with $O(n^{2.5}\log^3 n)$ runtime and an approximation factor of $O(\sqrt{n})$. This algorithm exploits the geometric structure of the problem, which is based on the idea of finding jointly heaviest monotone sequences (i.e., sequence of points with either increasing or decreasing order on each dimension) inside the auxiliary and target datasets. The main contributions of this paper are summarized as follows:

(1) To the best of our knowledge, we are the first to study data inference attacks against multi-dimensional OPE-encrypted databases by jointly considering all the dimensions simultaneously. We formulate a 2-D order-preserving matching problem and show its NP-hardness.

(2) We design two 2-D order-preserving matching algorithms, including a greedy and a polynomial time algorithm with approximation guarantees. We consider both unweighted and weighted cases, with different weight functions. We further explore efficiency enhancement using tree-based data structures. We also discuss extensions to higher dimensions. These algorithms have independent interest beyond the applications in this paper.

(3) We evaluate the efficiency and data recovery rate of our algorithms over both synthetic and real-world datasets for different application scenarios, including location-based services, census data, and medical data. Our results suggest that when the ciphertext dataset is highly similar to a subset of the plaintext dataset, the greedy min-conflict algorithm performs the best; but, in general, when these two datasets have arbitrary intersections and are less similar, our monotone matching algorithm performs better. Overall, the recovery rate of our 2-D algorithms significantly outperform single-dimensional matching algorithms when the data is sparse in each dimension.

2 BACKGROUND AND RELATED WORK

2.1 Order-Preserving Encryption

Order-Preserving Encryption (OPE) [27] is a special encryption, where the order of ciphertexts is consistent with the order of plaintexts. For instance, assume there are two plaintexts (m_1, m_2) and their OPE are ciphertexts $(\llbracket m_1 \rrbracket, \llbracket m_2 \rrbracket)$, where $\llbracket m_i \rrbracket$ is the encrypted version of m_i by following the common notations in previous studies [20, 27]. If $m_1 < m_2$, then $\llbracket m_1 \rrbracket < \llbracket m_2 \rrbracket$. With such property, comparison and sorting could be performed on encrypted data directly, without the need to access plaintext. While some OPEs are probabilistic and only reveal the order of data items [22], probabilistic OPEs increase the ciphertext size or require client-size storage, which scale poorly on sparse data. Most efficient OPEs are deterministic, and thus also reveal the frequency of data items [27]. In this paper, we focus on inference attacks on deterministic OPEs.

2.2 Inference Attacks on OPE via 1-D Matching

While the security of OPEs has been proved formally under Ordered Chosen-Plaintext Attacks [27], several studies propose inference attacks to evaluate the privacy leakage of OPE ciphertexts. For instance, Naveed, et al. [25] proposed an inference attack, named cumulative attack, on 1-D OPE by leveraging frequency leakage only. The authors address the attack by running the Hungarian algorithm. Grubbs et al. designed [20] leakage abuse attacks on 1-D OPE ciphertexts. The authors utilize both frequency and order leakage, and formulate the attack as a dynamic programming problem [20]. This leakage abuse attack performs faster than the cumulative attack and derives higher recovery rate. We briefly describe this leakage abuse attack below.

Given an OPE-encrypted dataset $A = \{\llbracket a_1 \rrbracket, \llbracket a_2 \rrbracket, ..., \llbracket a_n \rrbracket\}$ and an unencrypted dataset $B = \{b_1, b_2, ..., b_m\}$ similar to A, an attacker tries to infer the plaintexts of A without decrypting OPE ciphertexts, by leveraging the plaintexts of B as well as the order and frequency information of A and B. Without loss of generality, the attack assumes that A and B are sorted, where $\llbracket a_i \rrbracket < \llbracket a_j \rrbracket$ for any $i < j$, and $b_k < b_l$ for any $k < l$. The attacker also assumes $n \leq m$. Let $F_A(\llbracket a \rrbracket)$ and $F_B(b)$ be the Cumulative Distribution Function (CDF) of the OPE ciphertexts of dataset A and the plaintexts of dataset B respectively. Now, construct a bipartite graph H on vertex set A, B, in which the weight of an edge between vertex $[\![a_i]\!]$ and vertex b_j is defined as

$$w([\![a_i]\!], b_j) = \kappa - |F_A([\![a_i]\!]) - F_B(b_j)|$$

where κ is a pre-defined parameter and can be any integer greater than 1.

The attacker finds a max-weight bipartite matching in H that is (one-dimensional) order-preserving (i.e., a vertex early in A is mapped to an early vertex in B). Intuitively, suppose we plot the points of A and B on two parallel lines in their order. If we draw the edges in the matching, these edges could not cross. That is, if $[\![a_i]\!]$ and b_j are matched, any vertex in $[\![a_k]\!]$ with $k < i$ cannot be matched with vertex b_ℓ with $\ell > j$. Therefore, such a matching is also called a non-crossing matching. The max-weight non-crossing matching can be found in time $O(mn)$ via dynamic programming. If vertex b_j is matched with vertex $[\![a_i]\!]$, this attacker infers b_j as the plaintext of OPE ciphertext $[\![a_i]\!]$.

2.3 Other Attacks on Encrypted Databases

In addition to cumulative attacks and leakage abuse attacks, some other attacks have also been proposed against OPE. Durak et al. [14] proposed sort attacks on 2-D data encrypted by OPE. This attack performs a non-crossing matching on each dimension separately, and then improve the recovery results by evaluating inter-column correlation. Bindschaedler et al. [6] proposed an inference attack against property-preserving encryption on multi-dimensional data. This attack operates column by column. Specifically, it first recovers the column encrypted with the weakest encryption primitive, and then infers the next column encrypted by a stronger primitive by considering correlation. The attack is formulated as Bayesian inference problem. It also leverages *record linkage* and *machine learning* to infer columns that are strongly encrypted. In comparison, our proposed matching algorithms aim at optimally recover data tuples containing two or more dimensions as a whole. We utilize the order and frequencies of the 2-D tuples, instead of single-dimension order and frequency in previous works. In addition, we do not need explicit prior knowledge about the data correlations across dimensions within an encrypted dataset.

Finally, reconstruction attacks [16, 23] recover plaintexts on any searchable encryption that support range queries. Different from inference attacks, a reconstruction attack does not require a similar dataset as a reference but recover data based on access pattern leakage from a large number of range queries. However, reconstruction attacks often assume range queries are uniformly distributed, except [19], which is based on statistical learning theory. These works are orthogonal to this work.

In this paper, we design two 2-D order-preserving matching algorithms that jointly consider the data ordering on 2D. We also extend the 1-D matching algorithm in [20] to 2-D data for comparison. It turns out all the algorithms have advantages and limitations, as we describe in the evaluation and conclusion sections.

3 MODELS AND OBJECTIVES

System Model. In the system model, there are two entities, a *client* and a *server*. We assume that a client has a dataset (e.g., a location dataset) and needs to store it on the server. Due to privacy concerns, this client will encrypt the dataset before outsourcing it to the server.

We assume that the client encrypts the data using deterministic OPE, such that the server will be able to perform search operations (e.g., range queries) over encrypted data without decryption. We assume that each dimension of the data is encrypted separately with OPE, such that search can be enabled for each dimension. The client's data set is denoted as Q and its encrypted version as $[\![Q]\!]$.

Threat Model. We assume that the server is an *honest-but-curious* attacker, who is interested in revealing the client's data but does not maliciously add, modify, or remove the client's data. In addition, we assume that the server is able to possess a similar dataset P (in plaintext) as the client's dataset. In addition, we assume that P and Q have a significant common data points. For those points in Q that are also contained in P, they have similar frequency distributions. For example, Q can be the location data from Uber users, and P can be a USGS spatial database (Q can be considered to be randomly sampled from P). Or P and Q can be two location check-in datasets from two different social networking apps with partially overlapping locations.

Objectives. The attacker's goal is to perform inference attacks to maximally infer/recover the plaintext of encrypted database $[\![Q]\!]$ without decryption, using only $[\![Q]\!]$ and P with the ciphertext/plaintext order, either with or without frequency of points in both datasets. He aims at recovering the database points exactly. We define the *recovery rate* as the primary metric to measure the privacy leakage of the inference attack.

Recovery rate: If an attacker infers n points, m' of which are correct inference (the same as their true plaintext points), then the recovery rate is m'/n. In addition, we consider both the *unweighted* version of the above metrics, where each unique point/location is counted once, or the *weighted* version where the frequency is considered as well (number of 'copies' of the same point, e.g. the number of customers in a restaurant). The former can be regarded as "point-level" and the latter is "record-level". Intuitively, to maximize the weighted recovery rate, the points with larger frequencies should be correctly matched with high priority.

4 2-D ORDER-PRESERVING MATCHING

We formulate an order-preserving matching problem in two dimensions. Let P and Q be two finite sets of points in the plane. $P = \{p_1, p_2, \cdots, p_n\}$ and $Q = \{q_1, q_2, \cdots, q_m\}$. If $p \in P$ is matched to $q \in Q$, we denote it as an edge (p, q) and sometimes also denoted as $p \leftrightarrow q$. We say that a matching $\mathcal{M}$ between P and Q is *order preserving* if there exist two monotone functions ψ, ϕ such that if $(p, q) \in \mathcal{M}$ (for $p \in P, q \in Q$) then $q.x = \psi(p.x)$, $q.y = \phi(p.y)$.

It is convenient to consider an alternative, equivalent way to define order preserving, in terms of "conflicts". We say that two edges $(p, q) \in \mathcal{M}$ and $(p', q') \in \mathcal{M}$ are *in x-conflict with each other* if the x-projection (interval) of one edge contains the x-projection (interval) of the other edge; the notion of being in *y-conflict* is defined similarly. We say that a matching $\mathcal{M}$ is a *non-conflicting* *matching* of P and Q if it does not contain any x-conflicting or y-conflicting pair of edges. From the definitions, it is easy to see that a matching $\mathcal{M}$ is order preserving if and only if it is a non-conflicting matching.

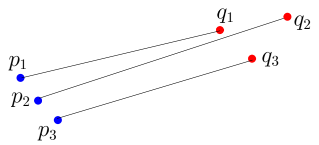


Figure 1: *In this matching $\mathcal{M}$ between $P = \{p_1, p_2, p_3\}$ and $Q = \{q_1, q_2, q_3\}$, the edge (p_2, q_2) is in y-conflict with edge (p_1, q_1) and in x-conflict with edge (p_3, q_3).*

We say that a point p' *dominates* p'', and write $p'' \prec p'$, if either (i) $p''.x < p'.x$, or (ii) $p''.x = p'.x$ and $p'.y < p''.y$. With this notation, two pairs (p_i, q_j), $(p_{i'}, q_{j'})$ with $p_{i'} \prec p_i$ but $q_j \prec q_{j'}$ are in *conflict.*

4.1 Unweighted v.s. Weighted Version

In this paper we study the problem of finding a maximum cardinality, or a maximum-weight order preserving matching.

In the unweighted version, we maximize the number of edges in a non-conflict matching between P and Q. This formulation does not use information on data frequencies.

To incorporate knowledge on data frequencies from P and Q, we can define weight of matching a point in P with a point in Q and ask for the non-conflict matching with maximum weight. The goal is to minimize the total difference of the frequencies between each ciphertext and its matched plaintext points. Note that this may or may not be equivalent with the objective of maximizing the recovery rate. This depends on the similarity of the two datasets P and Q: when the frequencies of the same points are close in either dataset, max-weight matching will likely maximize recovery rate.

There are several possible choices of weight function. Assume $f(p_i)$, $f(q_j)$ are the frequencies of locations (resp.) $p_i \in P, q_j \in Q$. Then the weight of matching p_i to q_j could be one of the following weight function:

(1) *weight*$(p_i \leftrightarrow q_j) = \min\{f(p_i), f(q_i)\}$. The rational for this weight function is that if we consider $f(p_i)$ and $f(q_j)$ as indicating the normalized number of items at point p_i and q_j, then $\min\{f(p_i), f(q_i)\}$ indicates the maximum number of items could be matched.
(2) *weight*$(p_i \leftrightarrow q_j) = \kappa - |f(p_j) - f(q_j)|$, where κ is a manually-picked constant, usually as maximum of all $f(q_i)$ and $f(q_j)$. This is the cost function used in [20].

4.2 Integer Programming Formulation

Given two sets of points, P and Q, we define a variable x_{ij} that takes value 1 if $p_i \leftrightarrow q_j$ and 0 otherwise. Now, we can formulate our matching problem as follows:

$$\begin{aligned}
\textbf{Maximize} \; & \sum_{i,j} x_{ij} \cdot w(p_i \leftrightarrow q_j) & \\
\textbf{Subject to} \; & \sum_{j} x_{ij} \le 1, & \forall i \\
& \sum_{i} x_{ij} \le 1, & \forall j \\
& x_{ij} + x_{i'j'} \le 1, & \forall (i,j), (i',j') \\
& \textbf{s. t.} \;\; (p_i, q_j) \text{ is in conflict with } (p_{i'}, q_{j'}). &
\end{aligned}$$

The first two constraints imply that one point can only be matched to one other point. The last inequality is the non-conflicting constraint.

4.3 Related Results on Maximum Independent Sets

Our problem can be phrased as a (weighted) maximum independent set (MIS) problem, in the conflict graph, defined below.

Conflict Graph $G_{\text{Conf}}(P \times Q, E_{\text{Conf}})$: the graph whose nodes are pairs of potentially matched points, one from P and one from Q, and whose edges represent the conflict relationship: $(u, v) \in E_{\text{Conf}}$ if the matched point pair $u \in P \times Q$ is in conflict with the matched point pair $v \in P \times Q$.

Unfortunately, this graph in our settings is enormous, and its node set has cardinality quadratic in the size of the input. Thus, pursuing our problem as a maximum independent set problem is likely impractical. In general, MIS has no polynomial-time constant factor approximation algorithm (unless $P = NP$); in fact, MIS, in general, is Poly-APX-complete, meaning it is as hard as any problem that can cannot be approximated within a polynomial factor [3]. However, there are efficient approximation algorithms for restricted classes of graphs. In planar graphs, MIS can be approximated to within any approximation ratio $c < 1$ in polynomial time; MIS also has a polynomial-time approximation scheme in any family of graphs closed under taking minors [18]. In bounded degree graphs, effective approximation algorithms are known with approximation ratios that are constant for a fixed value of the maximum degree; for instance, a greedy algorithm that forms a maximal independent set by, at each step, choosing a minimum-degree vertex in the graph and removing its neighbors, achieves an approximation ratio of $(\Delta + 2)/3$ on graphs with maximum degree Δ [21]; hardness of approximation for such instances is also known [5], and MIS on 3-regular 3-edge-colorable graphs is APX-complete [2].

5 NP-HARDNESS

The problem of finding a maximum-cardinality order preserving matching (i.e., the unweighted case) is NP-hard. Therefore the weighted setting is also NP-hard.

We establish this by using a reduction from the problem Pattern Matching Problem for Permutations (PMPP) [9], which asks the following: Given a permutation $T = (t_1, t_2, \ldots, t_n)$ of the sequence $(1, 2, \ldots, n)$ and a permutation $S = (s_1, s_2, \ldots, s_k)$ of the sequence $(1, 2, \ldots, k)$, for $k \le n$, determine if there exists a subsequence, $T' = (t_{i_1}, t_{i_2}, \ldots, t_{i_k})$, of T of length k (with $i_1 < i_2 < \cdots < i_k$)

such that the elements of T' are ordered according to the permutation S, i.e., such that $t_{i_j} < t_{i_{j'}}$ if and only if $s_j < s_{j'}$. We map a PMPP input pair of permutations, (T, S), to a pair of points, (P, Q), in the plane: Specifically, P is the set $\{(i, t_i) : 1 \le i \le n\}$ of n points corresponding to the permutation T, and Q is the set $\{(i, s_i) : 1 \le i \le k\}$ of k points corresponding to the permutation S. It now follows from the definition of an order preserving matching, and the specification of the PMPP, that there exists an order preserving matching of size k between P and Q if and only if there is a subsequence T' of T of length k such that the elements of T' are ordered according to the permutation S. It follows that our (unweighted) order preserving matching problem is NP-hard.

Theorem 5.1. *Given two point sets $P, Q \subset \mathbb{R}^2$, it is NP-complete to decide if there exists an order preserving matching $\mathcal{M}$ of cardinality* $\min\{|P|, |Q|\}$ *between P and Q.*

6 ALGORITHMS

6.1 Greedy Minimum-Conflict Matching

In this heuristic, we create an order preserving matching $\mathcal{M} \subseteq P \times Q$ in a greedy manner. We start with $\mathcal{M}$ empty, and at each iteration we add to $\mathcal{M}$ the edge that has the minimum number of conflicted edges among all potential future edges that could be selected. This heuristic is reminiscent of the minimum-degree heuristic of Halldórsson and Radhakrishnan [21] that shows that similar heuristics provide a $(\Delta + 2)/3$ approximation for finding a maximum independent set in graphs having maximum degree Δ; however, in our setting, Δ might be $\Omega(|P||Q|)$, making this bound uninteresting.

Formally, define for $p \in P, q \in Q$

$$s(p,q) = \sum\{w(p',q') | (p',q') \text{ conflicts with } (p,q) \text{ but not with edges currently in } \mathcal{M}\}$$

and greedily select (p^*, q^*) to minimize $s(p, q)$. A straightforward algorithm computes $s(p, q)$ directly (in time $O(n^2)$) for each of the $O(n^2)$ candidate edges (p, q), in order to select each edge to be greedily added to $\mathcal{M}$. Overall, this is $O(n^4)$.

6.1.1 Unweighted Case.

6.1.1 Unweighted Case. Here, to expedite the algorithm to avoid the time $O(n^4)$ (per edge selected), we propose a weighted random sampling approach. We could find (p^*, q^*) in amortized time $O(1)$ per pair (p_i, q_j). This is done in two steps: We first compute for each p_i the number $n^{\nwarrow}(p_i)$ of point $p \in P$ above and to the left of p_i. Similarly we define $n^{\nearrow}(p_i)$, $n^{\searrow}(p_i)$, $n^{\searrow}(p_i)$ and $m^{\nearrow}(q_j)$, $m^{\searrow}(q_j)$, $m^{\searrow}(q_j)$. Then the number of matching edges that are in conflict with (p_i, q_j) can be computed by evaluating the products $n^{\square}(p_i) \cdot m^{\square}(q_j)$, where $\square$ is one of the 4 directions $\nearrow$, $\nwarrow$, $\searrow$, $\swarrow$. As easily observed, the number of conflicts is

$$\begin{aligned} s(p_i, q_j) &= n^{\nwarrow}(p_i) m^{\nearrow}(q_j) + n^{\nearrow}(p_i) m^{\nwarrow}(q_j) \\ &\quad + n^{\swarrow}(p_i) m^{\searrow}(q_j) + n^{\searrow}(p_i) m^{\swarrow}(q_j) \\ &= nm - \sum_{\square \in \nwarrow, \nearrow, \swarrow, \searrow} m_i^{\square} n_i^{\square} \end{aligned}$$

We pick the edge minimizing this expression. Of course, once one edge is picked during the greedy matching algorithm, these numbers need to be recomputed, since multiple edges are not valid anymore.

We note that after the first iteration, when partial matching $\mathcal{M}$ is not empty, the values of $n_i^{\square}, m_j^{\square}$ reflects only edges not violating edges of $\mathcal{M}$. However, computing these values for every p_i, q_j in time $O(n^2)$ is straightforward.

6.1.2 Weighted case. We propose two basic methods.
Random sampling: We consider all n^2 potential edges, $P \times Q$, compute the weight of each, and pick a random sample R of (expected) size k, where the probability of picking (p, q) is

$$k \frac{w(p,q)}{\sum_{p',q'} w(p',q')}$$

Next, we greedily find a min-violation edge, with the violation computed with respect to R only. So the expected running time for this stage is $O(n^2 k^2)$ per edge added to $\mathcal{M}$.

This method can be enhanced further, for weight function (1), where the weight function is computed with respect to a random sample of *vertices* picked according to their weight.

ε-approximation via scaling algorithm. For the weight function (1) ($w(p_i, q_j) = \min\{f(p_i), f(q_j)\}$), a faster approach is proposed.

Let $w_{\min}, w_{\max}$ be the minimum weight and maximum weight. Consider the logarithmic number of levels $\{w_{\min}(1 + \varepsilon)^i\}$ for every i such that $w_{\min}(1 + \varepsilon)^i \le w_{\max}$. At the i'th step, we consider only the vertices with weight $\ge w_{\min}(1+\varepsilon)^i$, find the number ζ_i of edges conflicting (p, q) using the unweighted $O(n^2)$ algorithm, and sum the (rescaled) values $\sum(1 + \varepsilon)^i \zeta_i$ as an estimation of $s(p, q)$. It is easy to see that an (unscaled) edge conflicting (p, q) will be counted once, with its weight error bounded by a factor of $(1 + \varepsilon)$.

6.2 Greedy via Monotone Sequences

Given P, we say that a sequence $(p_1, p_2, , , p_k)$ is an ***monotone increasing sequence*** if $p_i.x \le p_{i+1}.x$ and $p_i.y \le p_{i+1}.y$, for all i. Then, a subset $P' \subseteq P$ is said to be a monotone increasing subset if the sequence obtained by ordering P' by x-coordinates is a monotone increasing sequence. Analogously we define sequences and subsets that are ***monotone decreasing***.

In the previous section we discussed methods to greedily augment the matching by a single edge. One might wonder if it is possible, and whether it is more efficient, to add a collection of edges at each time. For example, Dynamic Programming proved useful in the 1-D case, and it is tempting to apply it for the 2-D case as well. However, applying similar techniques for the 2-D case seems very challenging. It is extremely hard to define sub-problems which are independent on each other, in the sense that the solution of one does not depend on the solution to another. However, with a non-trivial hint on the approximation we obtain, we could define such a solution for monotone sequences. Refer to Algorithm 1 for the pseudo-code.

Essentially, if we opt to match $p \in P$ to $q \in Q$, then any decisions taken on the quadrant below and to the left of p could not (in an order-preserving matching) affect matching in the quadrant *opposite* this quadrant, consisting of points above and to the right of p. Similar observations hold for every pair of opposite quadrants. This observation suggests our search for *monotone sequences.*

Formally a sequence $P^{\uparrow} = \{p_1 \dots p_k\} \in P$ is an *increasingly monotone* sequence if $p_i \prec p_{i+1}$. (for $i = 1 \dots k-1$). Decreasing sequences are defined analogously. Obviously if in a matching $\mathcal{M}$, p_i is matched to $q_i \in Q$ then the sequence $\{q_1 \dots q_k\}$ is increasingly monotone as well. The *heaviest monotone sequence* is a monotone sequences maximizing the sum of weights of its edges. Given a partial matching $\mathcal{M}$, we describe in this section an algorithm that finds monotonic sequences $P^{\uparrow} \subseteq P$ and $Q^{\uparrow} \subseteq Q$, a matching between them that does not conflict with $\mathcal{M}$ and is of maximum weight. We use this algorithm as follows: In iterations we find an optimal monotonically (increasing or decreasing) sequences with respect to $\mathcal{M}$, include the corresponding matched edges to $\mathcal{M}$ and continue. Therefor we concentrate on efficient implementation of finding a single monotone matching. We discuss the case of monotonically increasing sequences. The case for monotonically decreasing sequences is handled analogously.

Let $\mathcal{X} = P \times Q = \{(p_i, q_j) | p_i \in P, q_j \in Q\}$. By abusing notation, we also consider each (p_i, q_j) as a point in $\mathbb{R}^4$, with coordinates $(p_i.x, p_i.y, q_i.x, q_i.y)$. We first describe the algorithm when $\mathcal{X}$ lies in $\mathbb{R}^4$, and then show that we could orthogonally project $\mathcal{X}$ into $\mathbb{R}^3$, and handle all querie as orthogonal three-dimensional range queries. For a point $p_i \in P$ we define $P_{\prec p_i} = \{p \in P \mid |p \text{ is dominated by } p_i\}$. For $(p, q), (p', q') \in \mathcal{X}$ we say that (p', q') dominates (p, q), and write $(p, q) \prec (p', q')$ iff $p \prec p'$ and $q \prec q'$. Similarly for $\mu = (p, q)$ (for $p \in P, q \in Q$), we write $w(\mu)$ to denote the weight of the matching edge (p, q). Fix $\mu = (p_i, q_j)$. We define $c[\mu]$ to be the maximum sum of weights of edges in any maximum increasing monotone matching by using only points of $P_{\prec p_i}$ to points of $Q_{\prec q_j}$ and ending at μ.

To obtain a fast asymptotic running time, we will use Algorithm 1. We maintain a 4-D orthogonal range tree $\mathcal{T}(\mathcal{X})$ [13]. Each leaf in the tree is associated with a node in $\mathcal{X}$. Each internal node $\eta \in T$ is associate with

(1) A range R_η which is a rectangle in $\mathbb{R}^4$.
(2) A subset $\mathcal{X}_\eta \subseteq \mathcal{X}$ which include all points of $\mathcal{X}$ inside R_η.
(3) The point $\mu^* \in \mathcal{X}_\eta$, which is the last point of the heaviest monotone sequence ending at μ^*, for $\mu^* \in \mathcal{X}_\eta$.
(4) $c(\eta)$ — the weight of this sequence.

The idea is to use an Orthogonal Range search data structure for the points in $\mathcal{X}$. We scan these points in topological increasing order, so if $\mu \prec \mu'$ then we access μ' after accessing μ. This will guarantee that $c(\mu)$ is fully computed at this point.

Lemma 6.1. *We could preprocess $\mathcal{X}$ into a data structure $\mathcal{T}$ such that the preprocessing time and space are both $O(n^2 \log^4 n)$, given a query axis-parallel rectangle $R \subseteq \mathbb{R}^4$, we could find a set of $O(\log^4 n)$ nodes $\Xi = \{\eta_1 \dots \eta_k\}$ of $\mathcal{T}$, each corresponds to a subset $\mathcal{X}_{\eta_i} \subseteq \mathcal{X}$ that is fully contained in R, and each is associated with a value $c[\mu_i]$ which is* $\max\{c[\mu'] \mid \mu' \in \mathcal{X}_{\eta_i}\}$

Next we notice that filtering points of $\mathcal{X}$ based on their first coordinate is not necessary. That is, we only need to store each point of $(p, q) \in \mathcal{X}$ using only $(p.y, q.x, q.y)$, since a query on other regions yields that the result is zero, and will not effect the query time nor the correctness.

Algorithm 1: Finding heaviest increasing monotone chain

```
1  Input: $P$ and $Q$ (sorted in an increasing order), and a partial matching
     $\mathcal{M} \subseteq P \times Q$ ;
2  Init: Initialize the an orthogonal range search $\mathcal{T}$ for $\mathcal{X}$,
3  and set $c(\eta) = 0,\ \forall \eta \in \mathcal{T}$.
4  for $i = 1$ to $|P|$ do
5      for $j = 1$ to $|Q|$ do
6          if $\mu = (p_i, q_j)$ does not conflict any edge in $\mathcal{M}$ then
7              Set the range (rectangle)
8              $R = \{(-\infty, p_i.y) \times (-\infty, q_i.x) \times (-\infty, q_i.y)\}$
9              Perform a range query in $\mathcal{T}$ with the range $R$ to obtain a
                 set $\Xi = \{\eta_1 \dots \eta_k\}$ of $O(\log^3 n)$ nodes in $\mathcal{T}$
               /* Each $\eta_i \in \Xi$ corresponds to a region fully
                   contains in $R$.                                    */
10             Let $\mu^*$ be $\arg\max\{c(\eta) | \eta \in \Xi\}$
11             Set $c(\mu) = w(p_i, q_j) + c(\mu^*)$
12             for each $\eta'_i \in \mathcal{T}$ ancestor of $\mu$ do
13                 Set $c(\eta'_i) = max\{c(\eta'_i), c(\mu)\}$
14             end
15         end
16     end
17 end
```

Lemma 6.2. *So the data structure is in $\mathbb{R}^3$. Hence the query time of Lemma 6.1 is improved to $O(\log^3 n)$. The space requirement also drops to $O(n^2 \log^2 n)$.*

Proof. Since all weights are positive, and the points of P are accessed in increasing x order lexicographically, then once accessing $c((p_i, q_j))$, its value is strictly positive only due to a point $p' \in P$ such that $p'.x \le p_i.x$. Therefor there is no need to filter nodes of $\mathcal{T}$ based on their very first coordinate. □

While orthogonal range trees are almost optimal theoretically, they suffer from several drawbacks. The space required is superlinear, and in practical applications, they tend to be inferior to other hierarchical spatial data structures as kD-trees. The latter could be applied with linear memory, and faster search time on realistic data.

Lemma 6.3. *Instead of the orthogonal range tree, if we use a 3-dimensional kD-tree, the space requirement will be linear, while the asymptotic running time per a query will increase to $O(|\mathcal{X}|^{(1-1/d)}) = O((n^2)^{2/3}) = O(n^{4/3})$.*

Running time and correctness . Given a partial matching M, it takes $O(n^2 \log^3 n)$ to find the heaviest (max-weight) monotone matching not conflicting M. At this point these edges are added to M, and the process repeats. Since P could be decompose into $\le \sqrt{n}$ monotone sequences [15], the number of iterations is $\le \sqrt{n}$. The overall running time is $O(n^{2.5} \log^3 n)$, and the space is $O(|P||Q| \log^2 n)$. Here $n = \max\{|P|, |Q|\}$.

6.3 Lower bounds

It is interesting to note that improving the bound below $\Omega(n^2)$ is unlikely, given that even if the points are on a line, then our problem is quite similar to the edit distance problem, and LCS problems, for which recent lower bounds are proven under the SETH assumption [1]. Hence we are only logarithmic distance away from the claimed optimum.

6.4 Approximation guarantees

Lemma 6.4. *Let opt be the maximum weight of the maximum order-preserving matching. Then by using the monotone sequences algorithm we obtain a matching of weight* $\geq opt/\sqrt{\min\{|P|,|Q|\}}$.

Proof. Consider any partition of P into monotone sequences. By the classical results of Paul Erdös and George Szekeres [15] $t \leq \sqrt{n}$. Let M^* be the set of all these edges, and let $M^*|_{P_i^{\uparrow}}$ be the set of edges with one endpoint in $P_i^{\uparrow}$ (for every $1 \leq i \leq l$). Since the sum of all matched edges in M^* is opt, the sum of all edges $M^*|_{P_i^{\uparrow}}$ must be $\geq opt/l$. This cost is obviously not larger than the maximum we find along any monotone sequence. □

6.5 Extension to Multi-Dimensional Data

Extending our matching algorithms to handle three or more dimensional data is well-motivated. Many databases have multiple columns of numerical data. For example, in the census database, there may be age, salary, zip code, etc. The data becomes increasingly sparse as the number of dimensions gets larger. Matching every single column separately will yield far from optimal results.

All our algorithms are still valid in this setting, where P and Q are points in $\mathbb{R}^d$ for a constant d, $d \geq 3$. However, the (worst-case) guarantees and running time of the monotone-greedy algorithm degrade. Since the longest monotone sequence of a set of n points in $\mathbb{R}^d$ is $\Theta(n^{1/d})$, we are only guaranteed a matching of weight $\geq opt/n^{1/d}$, where opt is the optimum weight order-preserving matching.

7 EVALUATION

In this section, we use both synthetic data and real-world datasets to evaluate the performance of our matching algorithms and attack effectiveness.

7.1 Data Sources

Synthetic Data Generation. The synthetic data is generated in the following manner. Let R be a set of uniformly distributed points in a 2-D area. Here R is a superset of the data. Then, for every $r \in R$ assign a point weight $f'(r)$, which is a uniformly distributed pseudo-random integer on the interval $[f_{min}, f_{max}]$. Let P denote the auxiliary/plaintext dataset and let Q denote the target/ciphertext dataset. P and Q are generated as follows.

For the case of Q being a subset of P: copy R into P, and then generate Q from R with the following sampling process. For each $r \in R$, we randomly and independently copy it as a point $q \in Q$ with probability $\beta \in (0,1)$. And if r was copied to a point $q \in Q$, then the frequency $f'(q)$ is assigned to be a binomial variable with probability p_{bion} and expectation $f'(r) \cdot p_{\text{bion}}$. For the other case that P intersects with Q, we sample the points of both P and Q from R randomly and independently with probability β, and similarly the weight of copied points are both sampled from a binomial distribution with probability p_{bion}. Finally, the integer point weights are normalized to frequency for P and Q in both cases, e.g. $f(p) = \frac{f'(p)}{\sum_{p' \in P} f'(p')}, \forall p \in P$.

The rationale of point frequency following a binomial distribution is that, in the real-world, the set of people who appear in one dataset may choose to be present or not in another dataset independently at random, e.g., Uber users can be regarded as randomly sampled from a USGS/census location database.

Real-world Data Sources. We use three real-world datasets to evaluate the performance of algorithms, and we start with the location check-in data from the social networking application Brightkite [11]. The location coordinates are expressed in latitude and longitude. We extract the records in a certain area (latitude: [37.700887, 37.826664], longitude: [-122.512317, -122.386762]), and take the data collected from Apr. 2008 to Apr. 2009 as the auxiliary (unencrypted) dataset while the data from Sep. 2009 to Sep. 2010 as the target (encrypted) dataset. We randomly choose 500 and 300 points from the reference and target datasets as P and Q to perform the matching. Then we process the data in P and Q by discretizing it into location grids whose granularity is tunable, which means we reserve certain number of digits after the decimal point to represent that grid. For example, if we reserve three digits (0.001 as unit grid length which is approximately 0.1 km in the real-world), there are 399 and 247 points in auxiliary and target datasets respectively. Note that we do not need to actually encrypt the target dataset in our evaluation since it does not change the matching result, as we only use the order and frequency information of the points.

We also use the city and town population totals for the US estimated by the United States Census Bureau [10]. To get the location information, we choose the 313 cities with latitude and longitude provided by Wikipedia [30] (excluding Jurupa Valley), associating with 2010 census data as the auxiliary dataset P. And we sample 180 cities, combining with the corresponding 2018 estimated population data to get the target dataset Q. Each city record has two dimensional location data, namely x and y coordinates representing latitude and longitude, with its population as frequency.

In addition to location datasets, we also evaluate our attacks in the context of medical data. We leverage a patient discharge dataset, which contains the distribution of inpatient discharges by principal diagnosis group for each California hospital [26]. We randomly sample the records collected in 2009 and 2014 as the auxiliary dataset and target dataset. More specifically, we select the numerical facility ID as the x coordinate. For the y coordinate, we convert the categorical diagnosis result to numerical data by assigning a specific diagnosis group to a numeric number. The frequency of each facility-diagnose pair is the recorded number of patients. There are overall 448 facilities in the original dataset, we first sample 30 of them as the facilities in P, and then sample 20 out of these 30 samples as facilities in Q. Then we extract the diagnosis and number of patients of the selected facilities to get y coordinate and frequency. At the end, we obtain 425 records in auxiliary dataset P and 272 records in target dataset Q.

7.2 Evaluation Metrics

For a given ciphertext record $c = [\![m]\!]$ and its plaintext m, the attack algorithm matches c to a corresponding plaintext as m' which might be different from m. We say c is correctly matched iff $m' = m$. Note that, when P intersects with Q, for a ciphertext $c = [\![m]\!] \in Q$, its plaintext m may or may not be in the set P. Hence we define the record set of ciphertexts and plaintexts as Q_r and P_r by: $Q_r = \{c \in Q | c = [\![m]\!], m \in P\}$, $P_r = \{m \in P | [\![m]\!] \in Q\}$. Define $I_{ij} = 1$ if point p_i is correctly matched to point q_j, where (p_i, q_j) is a plaintext-ciphertext pair; $I_{ij} = 0$ otherwise. We will use the following metrics for performance evaluation:

(1) Point recovery rate: $\sum_{j=1}^{j=|Q|} I_{ij}/|Q|$.
(2) Record recovery rate (ratio of people in correctly matched locations): $\sum_{j=1}^{j=|Q|} f_j I_{ij} / \sum_{i=1}^{i=|Q|} f_i$.
(3) Normalized objective, defined as the objective of an algorithm divided by the optimal ILP solution, which is to evaluate how far the algorithm is from the optimal objective function in the ILP formulation.

Note that there is an upper bound (the best one can do) of the recovery ratio. In the case of Q is a subset of P, the upper bound is 1; in the case of Q intersects with P, the upper bound is: $|Q_r| / |Q|$ for metric (1), and $\sum_{i \in Q_r} f_i$ for metric (2). To remove the impact of the size of the intersection, we use *normalized recovery rate* by dividing the recovery ratios with its upper bound in this paper.

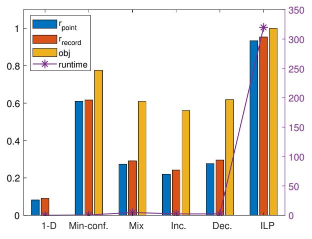
Besides, we found that the data density and the similarity of the frequency distributions for 1-D and 2-D data are two factors that have significant impact on algorithm performance, we define the following metrics to quantify them:

(1) Overlap ratio: the ratio of distinct records of ciphertext in Q to plaintext in P. For 2-D data, it is $|Q_r| / |P|$, for 1-D data, e.g. it is $|Q_{r_x}| / |P_x|$ on x-axis, where $|Q_{r_x}|$ and $|P_x|$ are the number of unique x coordinates in Q_r and P respectively. If we define the data density as the ratio of distinct points in Q against total points in the domain of all possible plaintexts, then the overlap ratio can be regarded as the *effective data density*, and when this ratio is small, the data is also sparse.
(2) Overall frequency similarity: denote the union of points in P and Q as R. For a point r in R, its frequency on P is defined as the frequency of the corresponding point if it appears in P, otherwise is 0. In this way, we get the frequency distribution of all these points on P and Q as $f(p) = \big(f(p_1), \cdots, f(p_k)\big)$ and $f(q) = \big(f(q_1), \cdots, f(q_k)\big)$. The overall frequency similarity of P and Q defined with Hellinger distance is:

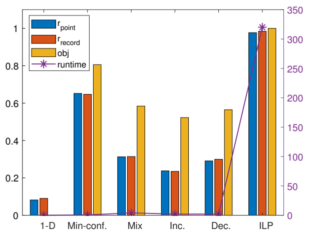
$$H\big(f(p), f(q)\big) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} \left(\sqrt{f(p_i)} - \sqrt{f(q_i)}\right)^2}$$

For Hellinger distance, we have $0 \leq H\big(f(p), f(q)\big) \leq 1$, and when $H\big(f(p), f(q)\big)$ is greater, the larger overall distance is between two distributions, and the less the similarity is.

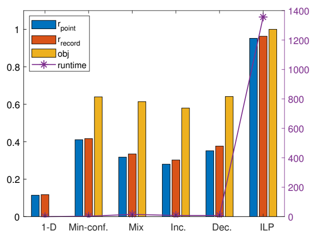
Also, we would like to compare the performance with a direct extension of the 1-D matching algorithm in [20] to 2-D data in the following way. We separately use the 1-D matching algorithm on each of the dimensions (x or y coordinates), with the order of x (or y) and marginal frequency of each x (or y) coordinate. For a point (c_x, c_y) in the target dataset, let m_x and m_y be the matched coordinates respectively, then we let (m_x, m_y) as the matched point for (c_x, c_y). Denote the point recovery rate of the 1-D matching on



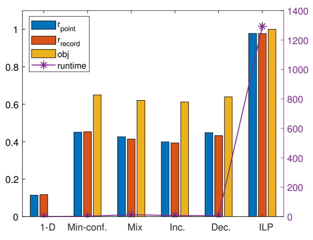
(a) Case 1, weight function 1



(b) Case 1, weight function 2



(c) Case 2, weight function 1



(d) Case 2, weight function 2

Figure 2: Synthetic data: point recovery rate (r_{point}), record recovery rate (r_{record}) and objective (bar, left y axis), and running time in seconds (line, right y axis).

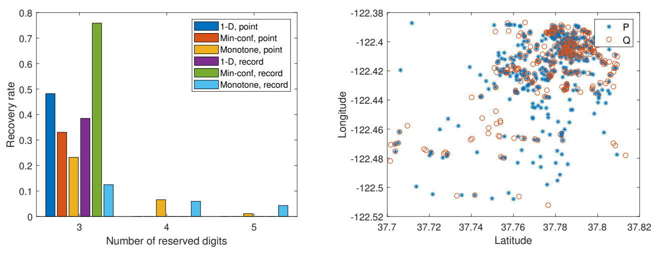
x and y coordinates as r_x and r_y, then for the point recovery rate on 2-D data r we have $r \leq \min\{r_x, r_y\}$.

7.3 Experimental Results for Synthetic Data

We implemented our proposed algorithms in Matlab, and all the experiments are run on a HPC cluster with 28 cores and 168GB memory. We first simulate with synthetic data, where we consider two cases: Q is a subset of P, and Q intersects with P. For each case, we generate 30 different datasets, and under each dataset, we evaluate the algorithm performance with the two weight functions defined in Sec. 4.1.

7.3.1 Synthetic Data, Case 1: Q is a subset of P. In this case, we set $P = R$, $|P| = 60$, Q is obtained from P with the data generation method described above, where $\beta = 0.6$, $p_{\text{bion}} = 0.7$. Results are shown in Fig. 2 (a)-(b), where "mix" means we find both increasing and decreasing monotonic matchings, and iteratively select the better one. We can see that the weight function 1 and 2 result in similar performance for all the 2-D algorithms, except that solution to ILP is the best (but also takes much more time, as shown in the right y axis). In this case the min-conflict algorithm outperforms all the other monotone sequences based algorithms under both weight functions. This is because the datasets are pretty similar. In addition, our 2-D matching algorithms significantly outperform the extended 1-D algorithm (about 10%) since our algorithms take into account the order of x and y axis simultaneously.

7.3.2 Synthetic Data, Case 2: P intersects with Q. In this case, we set $|R| = 100$, and both P and Q are obtained from R with the data generation method above, where β and p_{bion} are same as case 1. Besides, about 60% of points in P and Q are common. The average normalized recovery rates (point or record) of each algorithm among 30 runs are shown in Fig. 2 (c)-(d). Still, both weight functions



(a) (b)

Figure 3: Brightkite: (a) point and record recovery rate; (b) distribution of locations when reserve 4 digits

have similar performance. And also the monotone-sequence-based algorithm and the min-conflict algorithm have comparable performance. In addition, the normalized objectives are still close to that of the ILP solution (> 60%). Compared with case 1, in case 2 the normalized recovery rates are slightly lower for all the algorithms, since the two datasets are less similar and are more noisy.

7.4 Experimental Results on Real-world Datasets

We also evaluate our matching algorithms on the real-world datasets described at the beginning of this section. As the results in Sec. 7.3 show that the two weight functions have comparable performance, we only apply the weight function (2) on real-world datasets. For the monotone based greedy algorithms, we use either increasing or decreasing order, but only show the best results among them and refer it as the monotone solution.

7.4.1 Location-based Brightkite Check-ins. The recovery rate of point and record with different matching algorithms are shown in Fig. 3 (a), we can see that the finer the granularity is, the lower the recovery rate we get. And when we have a coarse granularity, e.g. when 3 digits reserved, the points on 1-D are much denser, and also the frequency distributions are more similar (as shown in Table 1), so the extended 1-D algorithm outperforms our 2-D algorithms. However, as the granularity becomes finer, the points are sparser and the frequencies are less similar, then the monotone matching algorithm performs the best. Note the lower recovery rate for finer granularity. This is because the algorithm relies on the frequency difference between two points, and both the ciphertext and plaintext's frequency distributions become more homogeneous under higher granularity, which increases the ambiguity in weighted matching (inclined to be unweighted). The solution to ILP should give us the actual optimal result, but unfortunately, since there are too many constraints in ILP, we cannot get its solution in time.

Table 1: Effective data density and Hellinger distance for Brightkite

Reserved digits	Effective data density			Hellinger distance		
	(x,y)	x	y	(x,y)	x	y
3	0.2807	0.8125	0.7283	0.5885	0.1786	0.2228
4	0.1869	0.4595	0.4053	0.6393	0.6837	0.4773
5	0.1740	0.2120	0.2	0.6493	0.6214	0.6340

7.4.2 City and Town Population. The matching results are given in Table 2, where the monotone matching yields a slightly lower recovery rate than the min-conflict algorithm, but better performance than the extended 1-D algorithm. Similarly, we compute the Hellinger distance (frequency similarity) and effective data density. It turns out that these two values on 2-D and 1-D data are the same, which are 0.5241 and 0.5751 respectively. This is because all the x and y coordinates are unique, so as to the (x, y) data. Hence, in this case for the extended 1-D algorithm and 2-D algorithms, the effects of the data density and frequency distribution are the same, the recovery rate here is decided by the advantage of the algorithm itself. The 2-D algorithms outperform the extended 1-D algorithm on inference against 2-D OPE databases because we jointly consider the data orders on x and y coordinates. And we use an extreme case in Fig. 4 to show the benefits of considering data orders in 2D.

Table 2: Results for city and town population dataset

Algorithm	Recovery rate		Runtime (s)
	Point	Record	
1-D	0	0	11.8830
Min-conf.	0.3778	0.3653	264.0576
Monotone	0.25	0.2631	309.6281

7.4.3 Patient Discharge Data by Principal Diagnosis. We show the results of patient discharge data in Table 3, where the monotone matching yields very high recovery rates. But it is surprising that the min-conflict algorithm yields zero recovery rates, even worse than the extended 1-D algorithm. First we observe that for the adopted dataset, the set of y coordinates of the auxiliary and target datasets are the same, which correspond to 19 principal diagnosis groups (hence the data density on y-axis is 1 in Table 4), and this results in an exact matching on y-axis for all the algorithms. Then the recovery rate is decided by the matching on x-axis, which means the 2-D order cannot benefit 2-D algorithms much. Hence in this case the noisy data frequency has a greater impact on the min-conflict algorithm, which leads to poor performance.

Table 3: Matching results for patient discharge data

Reserved digits	Recovery rate		Runtime (s)
	Point	Record	
1-D	0.0684	0.0290	0.249
Min-conf.	0	0	517.9117
Monotone	0.8441	0.8995	196.3670

Table 4: Effective data density and Hellinger distance for patient discharge data

Effective data density			Hellinger distance		
(x,y)	x	y	(x,y)	x	y
0.6188	0.6667	1	0.3798	0.1665	0.0960

8 CONCLUSION AND FUTURE WORK

In this paper, we studied the problem of inferring data from OPE-encrypted databases by jointly considering the multi-dimensional order and frequency of data tuples. We formulate it as a multi-dimensional matching problem, prove the NP-hardness, and propose efficient algorithms to solve it. Our algorithms exploit the geometric structure of the problem. We show that the monotone matching could be obtained in asymptotic time $O(n^{2.5} \log^3 n)$ that is comparable to the $O(n^2)$ 1-D algorithm. A simpler greedy algorithm is also provided. Experimental results on synthetic and real-world

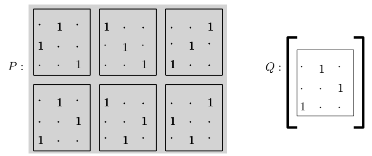


Figure 4: *1D-matching of [20] vs. 2D-matching. Most points in P have frequency* = 0*, or do not appear in Q. These elements are represented by dots. Each block in P is associated with a unique permutation of* (1, 2, 3)*. In Q, only one of these 6 blocks appear. Obviously the 2-D data uniquely indicates which blocks of P is isomorphic to Q, and (if only blocks are considered), this block is revealed as the unique optimal solution to the order-preserving matching. On the other hand, the sum of each row and column are identical, so no meaningful 1-D matching is possible. By replacing* 3 *by an arbitrary d, we indicate that no less than d*! *false matching are possible.*

datasets show that our algorithms perform better than the extended 1-D algorithm when the data is sparse. In addition, the performance gain depends on data density and similarity of frequency distributions, but 2-D matching is more robust to noise (or dissimilarities between the frequency distributions). Our algorithms shed more light on the security evaluation of OPE encrypted databases.

Extensions and Future Directions: Our experiments indicate that the 1-D matching algorithm is likely to *cluster* the data into meaningful regions, and match region in an order-preserving way, using the aggregated data in each region. We think clustering could average/smooth the impact of noise, downsize the size of the problem, and circumvent issues when the datasets use a different scale of coordinates, hence we would like to combine clustering with monotone matching algorithms. Clustering the plaintext P could be obtained based on geographic/geometric vicinity in a straightforward way, thus we assume it is given in the rest for simplicity. However, finding corresponding regions in the ciphertext Q is not obvious, as distances are distorted. Furthermore, finding these clusters cannot be decoupled from the matching process. We have obtained some preliminary results and we briefly describe them here. For any two rectangles R_1, R_2 the cost of matching the point of P inside R_1 to points of Q inside R_2 depends on the difference between their corresponding sum of frequencies in each rectangle. Our optimization function is to simultaneously obtain a clustering of Q and an one-to-one order-preserving matching to P's cluster. The 1-D problem can easily be solved in time $O(n^2)$, using dynamic programming. This leads to an $O(n^5)$ algorithm for the 2-D problem for matching monotonic set of points in P to a weakly-monotonic set of rectangles in Q. We omit details due to lack of space. Testing the effectiveness of these approaches is left for future work.

REFERENCES

[1] Arturs Backurs and Piotr Indyk. 2015. Edit distance cannot be computed in strongly subquadratic time (unless SETH is false). In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 51–58.

[2] Cristina Bazgan, Bruno Escoffier, and Vangelis Th Paschos. 2005. Completeness in standard and differential approximation classes: Poly-(D) APX-and (D) PTAS-completeness. *Theoretical Computer Science* 339, 2-3 (2005), 272–292.

[3] Cristina Bazgan, Bruno Escoffier, and Vangelis Th. Paschos. 2005. Completeness in standard and differential approximation classes: Poly-(D)APX- and (D)PTAS-completeness. *Theor. Comput. Sci.* 339, 2-3 (2005), 272–292.

[4] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. 2007. Deterministic and efficiently searchable encryption. In *Annual International Cryptology Conference*. Springer, 535–552.

[5] Piotr Berman and Marek Karpinski. 1999. On some tighter inapproximability results. In *International Colloquium on Automata, Languages, and Programming*. Springer, 200–209.

[6] V. Bindschaedler, P. Grubbs, D. Cash, T. Ristenpart, and V. Shamtikov. 2018. The Tao of Inference in Privacy-Protected Databases. In *Proc. of VLDB'18*.

[7] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'neill. 2009. Order-preserving symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 224–241.

[8] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. 2011. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *Annual Cryptology Conference*. Springer, 578–595.

[9] Prosenjit Bose, Jonathan F. Buss, and Anna Lubiw. 1998. Pattern Matching for Permutations. *Inf. Process. Lett.* 65, 5 (1998), 277–283. https://doi.org/10.1016/S0020-0190(97)00209-3

[10] United States Census Bureau. 2019. City and Town Population Totals: 2010-2018. (2019). https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?src=bkmk

[11] E Cho, SA Myers, and J Leskovec. 2011. Friendship and mobility: Friendship and mobility: User movement in location-based social networks. *Proc. ACM SIGKDD* (2011).

[12] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. 2011. Searchable symmetric encryption: improved definitions and efficient constructions. *Journal of Computer Security* 19, 5 (2011), 895–934.

[13] Mark De Berg, Marc Van Kreveld, Mark Overmars, and Otfried Schwarzkopf. 1997. Computational geometry. In *Computational geometry*. Springer, 1–17.

[14] F Betul Durak, Thomas M. DuBuisson, and David Cash. 2016. What Else is Revealed by Order-Revealing Encryption. In *Proc. of ACM CCS'16*.

[15] Paul Erdös and George Szekeres. 1935. A combinatorial problem in geometry. *Compositio mathematica* 2 (1935), 463–470.

[16] Georgios Kellaris and George Kollios and Kobbi Nissim and Adam O'Neil. 2016. Generic Attacks on Secure Outsourced Databases. In *Proc. of ACM CCS'16*.

[17] Google. 2018. Google Encrypted Bigquery Client. (2018). https://github.com/google/encrypted-bigquery-client

[18] Martin Grohe. 2003. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica* 23, 4 (2003), 613–632.

[19] P. Grubbs, M. Lacharite, B. Minaud, and K. G. Paterson. 2019. Learning to Reconstruct: Statistical Learning Theory and Encrypted Databse Attacks. In *Proc. of IEEE S&P'19*.

[20] Paul Grubbs, Kevin Sekniqi, Vincent Bindschaedler, Muhammad Naveed, and Thomas Ristenpart. 2017. Leakage-Abuse Attacks against Order-Revealing Encryption. In *Proc. of IEEE S&P'17*.

[21] Magnús M Halldórsson and Jaikumar Radhakrishnan. 1997. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. *Algorithmica* 18, 1 (1997), 145–163.

[22] Florian Kerschbaum and Axel Schroepfer. 2014. Optimal average-complexity ideal-security order-preserving encryption. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 275–286.

[23] Mari-Sarah Lacharite, Brice Minaud, and Kenneth G. Paterson. 2018. Improved Reconstruction Attacks on Encrypted Data Using Range Query Leakage. In *Proc. of IEEE S&P'18*.

[24] Microsoft. 2019. Microsoft Always Encrypted (Database Engine). (2019). https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/always-encrypted-database-engine?view=sql-server-2017

[25] M. Naveed, S. Kamara, and C. V. Wright. 2015. Inference Attacks on Property-Preserving Encrypted Databases. In *Proc. of ACM CCS'15*.

[26] State of California. [n. d.]. Patient Discharge Data by Principal Diagnosis. ([n. d.]). https://healthdata.gov/dataset/patient-discharge-data-principal-diagnosis

[27] Raluca Ada Popa, Frank H Li, and Nickolai Zeldovich. 2013. An ideal-security protocol for order-preserving encoding. In *2013 IEEE Symposium on Security and Privacy*. IEEE, 463–477.

[28] Raluca Ada Popa, Catherine Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*. ACM, 85–100.

[29] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. 2000. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. IEEE, 44–55.

[30] Wikipedia. 2019. *List of United States cities by population*. https://en.wikipedia.org/wiki/List_of_United_States_cities_by_population

[31] Wikipeida. 2019. Facebook–Cambridge Analytica data scandal. (2019). https://en.wikipedia.org/wiki/Facebook-Cambridge_Analytica_data_scandal

[32] Wikipeida. 2019. Yahoo! Data Breaches. (2019). https://en.wikipedia.org/wiki/Yahoo!_data_breaches