

Chapter 2

Privacy Preservation Techniques

In this chapter, we will introduce some basic concepts and fundamental knowledge of privacy preservation techniques. Privacy preservation has become a major issue across different applications, from data publishing to location-based services. Although the applications and scenarios are quite different, the essential privacy models and the core techniques are relatively the same. The state-of-the-art privacy techniques can be categorized into four classes: anonymization, perturbation, differential privacy, and cryptographic techniques. In location privacy protection, anonymization boils down into so-called spatial cloaking. Since anonymization and spatial cloaking adopt the same idea, we will discuss them in one section. Cryptography is a very general terminology and covers plenty of specific techniques. In this chapter, we only focus on private information retrieval, which is most related to database query in CRNs. A brief review on the literature of each category will be provided at the end of each section.

2.1 Anonymization and Spatial Cloaking

In this section, we will discuss some fundamental concepts used in anonymization and spatial cloaking. In order to get a better understanding of these privacy preservation techniques, we first illustrate them using a toy example.

2.1.1 Anonymization Operations

we consider six nodes in a network, and each node has the following three attributes

- Identifier (ID) that can uniquely identify a node.
- Longitude.
- Latitude.

Table 2.1 Location data in the example

Identifier	Location data	
	Longitude	Latitude
1	20	20
2	25	17
3	20	40
4	27	35
5	39	27
6	38	29

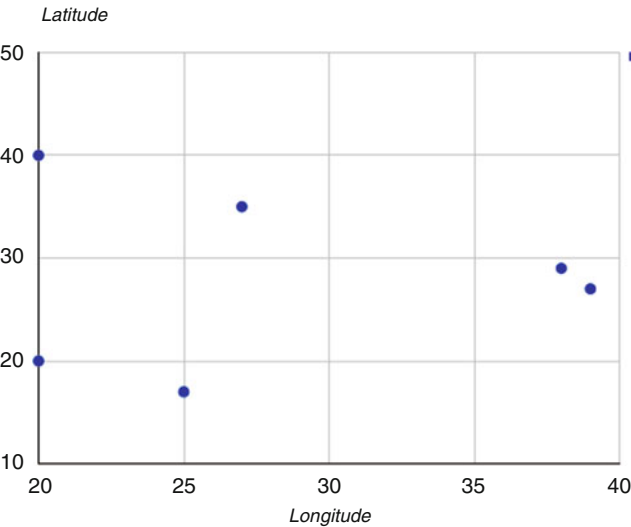


Fig. 2.1 Locations in the example

Longitude and latitude attributes together form a two-dimensional location data that describe the exact two-dimensional geo-location of a node. In a location sensitive scenario, the nodes do not want others to know their exact physical location.

Example 1. Table 2.1 shows an example of location data in this network.

Table 2.1 belongs to relational model of data, where each tuple (e.g., node) attaches a set of attributes (longitude and latitude). As a visual aid, the two-dimensional locations are depicted in Fig. 2.1.

In order to protect node’s location privacy, the nodes can *anonymize* or *cloak* their location data before sharing the data with other entities (e.g., the FC or the database). The core idea of anonymization or spatial cloaking consists of the following two steps

- Breaking, which is an operation that divides the nodes into a set of groups and then breaks the exact links between the identifiers and location attributes. Then, the one-to-one mapping between identifiers and locations are weakened by breaking the linkage so that the adversary has less confidence in inferring a node’s location. As such,

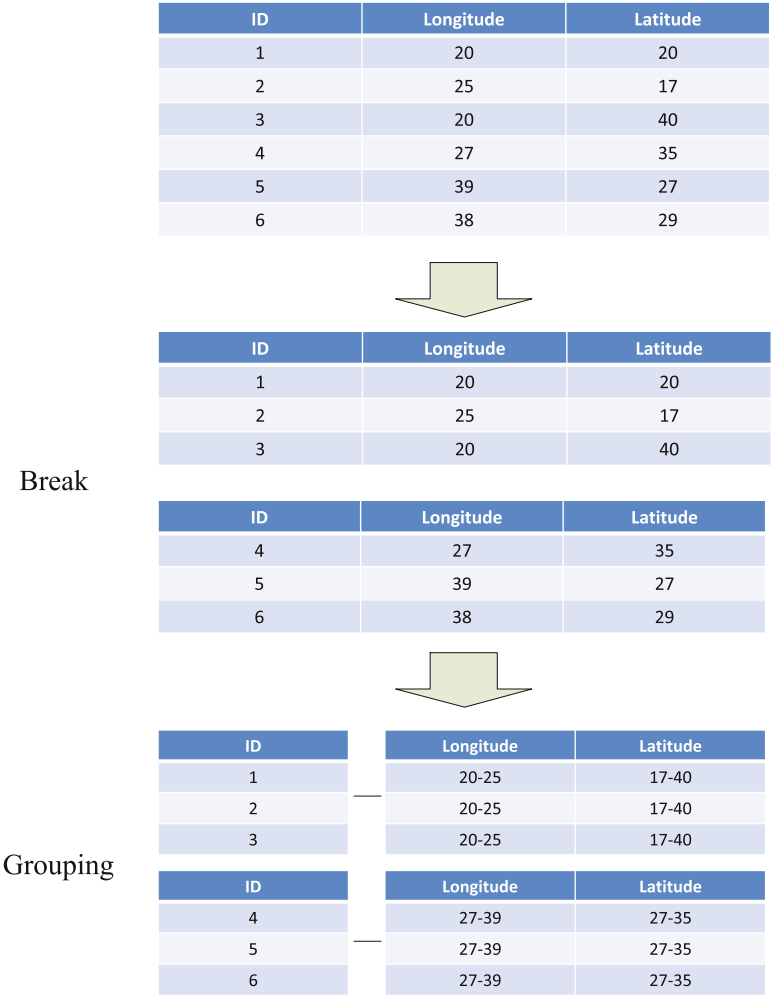


Fig. 2.2 Anonymizing the example

- Grouping. Via grouping, all nodes in the same group are indistinguishable in their locations. The most common way to group nodes is generalization, or so-called spatial cloaking in location privacy. Generalization is an operation changing an exact value to a more generalized one, e.g., a numeric value can be generalized to a range value.

Figure 2.2 describes the grouping-and-breaking and generalization operations on Table 2.1. We can see that the nodes are divided into two groups, and the location data in each group is generalized to identical ranges. A two-dimensional range can be viewed as a region or a cloak. The generalization operation to group exact location points into a region is referred to as spatial cloaking, are depicted in Fig. 2.3.

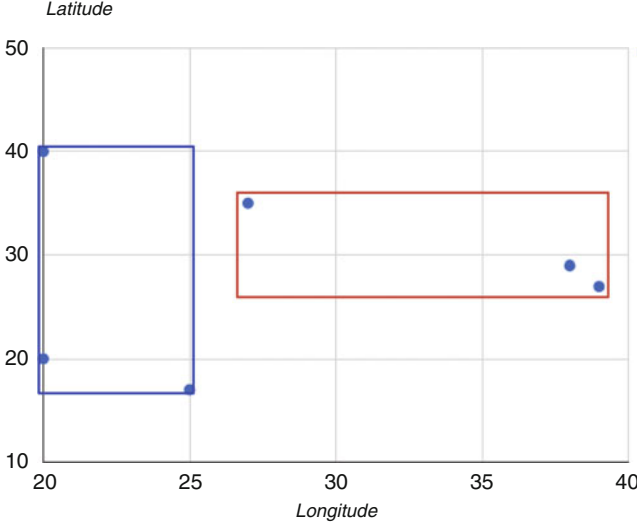


Fig. 2.3 Spatial cloaking over the example

As mentioned in introduction, location privacy consists of two categories: query privacy and location privacy. Here we use an example in LBS to show the difference of using k -anonymity to protect query privacy and location privacy.

Example 2. Similar to the above example, k -anonymity is applied to the location information contained in the query so that the exact locations of k users are generalized to the same region. In this way, the identity of each user is protected since all users are indistinguishable from each other by looking at the queries. Thus, query privacy is well protected. However, when the k users locate in a very small region, an adversary can locate the users in a restricted area without knowing the identity of each query. An extreme case is that all users co-locate together, i.e., all users stand in the same location spot. In such situation, the location privacy of the users are compromised.

In the above example we can see that even if k -anonymity is applied, the location privacy can still be compromised. To address this issue, location entropy, inspired by Shannon's entropy in information theory, is proposed to measure privacy level. Location entropy describes adversaries' uncertainty about a user's location. Normally, a set of points of interest (POI) is specified, and is usually assumed that the adversaries have equal inference beliefs of a user visiting a POI. Note that with equal beliefs, the location entropy is the largest, meaning that the adversaries' uncertainty is the highest before observing the location queries. The posterior belief entropy is used to measure the uncertainty about a user's location after observing the location queries.

A major limitation of location entropy is that the choice of POI affect the result of entropy. Another issue is that location entropy cannot tell how accurately an

adversary can infer a user's location. To cope with these issues, expected distance error is used as a improved version location entropy, where the location error of an inferred location is taken into consideration. The location error refers to the distance between the inferred location and the true location of a user. Combining inference beliefs, expected location error can be computed. As such, different POIs have different location error and thus weight different in the measurement.

The information loss in the process of anonymization is inevitable, which is a tradeoff for protecting privacy. To quantify the information loss caused by anonymization operations, several information metrics are developed. In the following part of this subsection, we will introduce several information loss metrics.

A simple measure on information loss is minimal distortion (MD), where a unit penalty is counted when each element in the data table is generalized or modified. For example, in Fig. 2.2, a total of 12 elements (i.e., 6 individuals, and each individual has 2 elements generalized) have been generalized, and thus, 12 units of distortion will be counted. MD is usually suitable for operations on categorical element, while for operations on numerical elements, information loss and discernibility metric are commonly used.

Information loss, or $ILoss$, first proposed in [108], is a metric capturing the information loss of generalizing a specific value to a general range

$$ILoss(R(v_A)) = \frac{R(v_A)}{D(A)}, \quad (2.1)$$

where v_A is an exact value of attribute A , $R(v_A)$ the generalized range of the value v_A , $D(A)$ the domain of the attribute A . $ILoss(R(v_A))$ measures the information loss caused by generalizing v_A to $R(v_A)$, i.e., the fraction of domain values generalized by $R(v_A)$. For instance, if the domain of a location data is $[1, 60]$, generalizing the location 5 to a range $[1, 10]$ has information loss of $(10 - 1)/60$. The total information loss of an individual u can be computed by

$$ILoss(u) = \sum_{A \in \mathcal{A}} (\omega_A ILoss(R(v_A))), \quad (2.2)$$

where ω_A specifies the importance of the attribute A , \mathcal{A} the set of all attributes. The overall loss can be measured by

$$ILoss(\mathcal{U}) = \sum_{u \in \mathcal{U}} ILoss(u), \quad (2.3)$$

where \mathcal{U} is the set of all individuals.

The normally used metric for k -anonymity is discernibility metric (DM) [52]. The aim of DM is to quantify the loss for each individual being anonymized to be indistinguishable from other individuals with respect to QIs. Let $|QI_i|$ denote the size of a QI group i . Then, the penalty of generalizing this group is $|QI_i|^2$. The overall penalty can be expressed as

$$ILoss(\mathcal{U}) = \sum_i |QI_i|^2. \quad (2.4)$$

2.1.2 Anonymization Privacy Models

In the previous subsection we discussed the fundamental operations in anonymization technique. In this subsection, we will introduce some anonymization privacy models to quantitatively analyze location privacy protection.

As the first and the most fundamental anonymization privacy model, k -anonymity [89] has been proposed to protect individual privacy in data publishing.

Definition 1 (k -Anonymity). In each group partitioned by a certain anonymization algorithm is said to satisfy k -anonymity if the size of the group equals or is larger than k . A table is said to satisfy k -anonymity if each group in the table satisfies k -anonymity.

The goal of k -anonymity is to ensure that each individual is indistinguishable from at least $k - 1$ other individuals in the table. In a location sensitive network, k -anonymity is satisfied if each node's shared location in the network is indistinguishable from at least $k - 1$ other nodes' locations. In Fig. 2.2, we can see that the table is 3-anonymity: locations of node 1, 2, 3 are generalized to the same region and thus cannot distinguish from each other, and so do locations of node 4, 5, 6.

Based on the notion of k -anonymity, many other anonymization models have been proposed. Here we introduce several popular anonymization models, including ℓ -diversity [66], (α, k) -anonymity [107].

The essential observation of ℓ -diversity is that the attributes of an individual consists of two parts: quasi-identifiers (QIs) and sensitive attributes (SAs). QIs refer to the attributes that can semi-determine the individual, and usually correspond to personal information such as gender, nationality, and age. QIs can be used to identify an individual. A previous study [89] shows that 87% individuals in a medical data set could be uniquely identified by gender, date of birth and zip code. These QIs are accessible from some publicly known data set; examples include a voting registration table. According to Sweeney [89], most municipalities sell the identifiers of individuals along with basic demographics, including local census data, voter lists, city directories, and information from motor vehicle agencies, tax assessors, and real estate agencies; the study [89] also points out that a city's voter list in two diskettes was purchased for only 20 dollars, and was used to identify medical records. QIs are usually considered to be publicly known and are treated as background knowledge of the adversaries. In location privacy, the longitude and latitude attributes can be considered as QIs since they can identify a user by its physical location.

SAs are the attributes that are private to the individuals, such as home address in a location application or disease attributes in a medical data set. Given QIs as background knowledge, k -anonymity may not be able to protect individual's privacy. Consider Table 2.1 again with one more attribute "AtHome" added. 'AtHome' is sensitive to all users, as described in Table 2.2. After applying anonymization algorithm on Table 2.2, we can see the k -anonymous table with sensitive attribute "AtHome" as depicted in Fig. 2.4. We can see that although the

Table 2.2 Location data with “AtHome” attribute in the example

Identifier	Location data (QI)		SA
	Longitude	Latitude	AtHome
1	20	20	Y
2	25	17	Y
3	20	40	Y
4	27	35	N
5	39	27	N
6	38	29	N

ID	Longitude	Latitude	AtHome
1	20-25	17-40	Y
2	20-25	17-40	Y
3	20-25	17-40	Y

ID	Longitude	Latitude	AtHome
4	27-39	27-35	N
5	27-39	27-35	N
6	27-39	27-35	N

Fig. 2.4 k -anonymous table with sensitive attribute

exact location of users are anonymized, we can tell whether a user is at home or not since in each group the values of the “AtHome” attribute are identical. To tackle this case, ℓ -diversity is proposed [66].

Definition 2 (ℓ -Diversity). In each group partitioned by a certain anonymization algorithm is said to satisfy ℓ -diversity if the probability that any individual in this group is linked to a sensitive value is at most $1/\ell$. A table is said to satisfy ℓ -diversity if each group in the table satisfies ℓ -diversity.

By definition, we can see that the example in Fig. 2.4 only satisfies 1-diversity. To improve the diversity of sensitive values, we anonymize the table with a different partition strategy as illustrated in Fig. 2.5. In Fig. 2.5, it can be seen that each sensitive value in a group has frequency no more than 2 and the group size is fixed at 3, thus, the result satisfies 2/3-diversity as well as 3-anonymity. Figure 2.6 depicts the spatial cloaking results.

An important property shared by the aforementioned two anonymization models is *monotonicity* property, which is commonly used as an essential observation to design anonymization algorithms. The definition of monotonicity is given as follows.

Definition 3 (Monotonicity). A privacy model \mathcal{M} is said to satisfy the monotonicity property, if for any two groups G_1 and G_2 that satisfy \mathcal{M} , the merged group $G_1 \cup G_2$ satisfies \mathcal{M} .

ID	Longitude	Latitude	AtHome
1	20-39	17-27	Y
2	20-39	17-27	Y
5	20-39	17-27	N

ID	Longitude	Latitude	AtHome
3	20-39	29-40	Y
4	20-39	29-40	N
6	20-39	29-40	N

Fig. 2.5 Another k -anonymous table ℓ -diversity considered

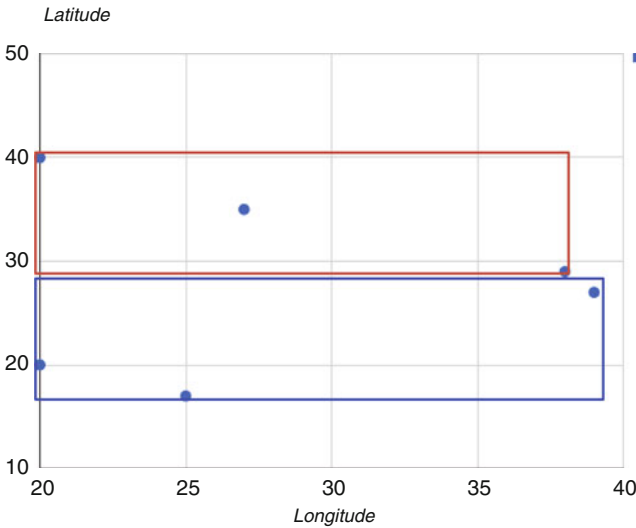


Fig. 2.6 Spatial cloaking with ℓ -diversity considered

Both k -anonymity and ℓ -diversity satisfy the monotonicity property. Taking Fig. 2.5 as an illustration. In Fig. 2.5, each of the two groups satisfies $2/3$ -diversity and 3-anonymity. If we merge the two groups into one single group containing all six individuals, the resulting group still satisfies $2/3$ -diversity and 3-anonymity.

Besides the above-mentioned anonymization models, there are many alternative models, most of which can be considered as the enhanced or variant versions of k -anonymity or ℓ -diversity.

Jointly considering the requirements of k -anonymity and ℓ -diversity, (α, k) -anonymity [107] is proposed. In (α, k) -anonymity model, α is a real number within the interval $[0, 1]$ and k is a positive integer. A table is said to satisfy (α, k) -anonymity model if the size of each group in the table is at least k and the frequency of each sensitive value in a group is at most αk . The interpretation of (α, k) -anonymity model can be the combination of ℓ -diversity and k -anonymity where α is set to $1/\ell$.

A variant version of ℓ -diversity is entropy ℓ -diversity, where instead of using frequency to bound sensitive values, entropy is adopted.

$$-\sum_{s \in \mathcal{S}} \Pr(s) \log \Pr(s) \geq \log(\ell), \quad (2.5)$$

where s is a sensitive value, \mathcal{S} the set of all possible values of a sensitive attribute, $\Pr(s)$ the fraction of individuals in a group with sensitive value s . The left hand side of the equation is called the entropy of the sensitive attribute. The purpose of entropy ℓ -diversity is to generate a result with more evenly distributed sensitive values in each group.

To cope with high-dimensional data table, Mohammed et al. [76] reported that in real-life attacks, the adversaries could hardly acquire all QIs. Based on this observation, Mohammed et al. proposed (LKC)-privacy, where the prior knowledge of adversaries is assumed to be limited to at most L QI attributes, K and C are the bounds similar to k -anonymity and ℓ -diversity.

The notion of t -Closeness is proposed [63] to maintain the overall distribution of a sensitive attribute. Consider a case where 95% of individuals are in a coffee shop area while only 5% are scattered elsewhere. Suppose a group with 50% of individuals in a coffee shop and 50% elsewhere, and therefore satisfies 2-diversity. However, this group leaks extra information on location spots since any individual in this group could be inferred as being elsewhere with much higher confidence compared with overall probability of being elsewhere. To prevent such information leakage, t -Closeness [63] requires the distribution of a sensitive attribute in any group to be close to the distribution of the attribute in the overall data set. Earth Mover Distance (EMD) function is leveraged to measure the closeness between two distributions, and the closeness is bounded to be within t .

2.2 Random Perturbation

Random perturbation [4, 105] is a popular method for eliciting information from individuals without compromising privacy, which has been adopted in many privacy preservation applications such as advertisement targeting [57], data mining [33, 109], collaborative sensing [65], collaborative spectrum sensing [64] due to its simplicity, efficiency, and statistical preservation. The basic idea of random perturbation is to replace the original data values with some synthetic data values so that the statistical information remains relatively the same while the original values never get disclosed. Due to the randomization, an individual perturbed value can be quite different from its original value. As such, the true values are kept private and cannot be inferred by the adversaries by linking private attributes to a certain individual.

There are many different random perturbation techniques. First, we illustrate the core idea of random perturbation by introducing a specific random perturbation

which is referred to as uniform perturbation. The principle of uniform perturbation is as follows: for each sensitive value v , we toss a coin with head probability of p and tail probability of $1 - p$. If the tossing result is head, v remains unchanged; otherwise v is replaced with a random value sampled from its domain. For example, $v = 3$, $p = 0.6$, the domain of v is integers within $[1, 10]$, then the mapping result of v , denoted as \hat{v} , can be written as

$$\hat{v} = \begin{cases} v & \text{with probability } 0.6 \\ X & \text{with probability } 0.4 \end{cases} \quad (2.6)$$

where X is a random variable follows a discrete uniform distribution that each value in the domain is chosen with equal probability. Then, the probability that v is retained is $0.6 + 0.4 \times 1/10 = 0.64$. The probability mass function (pmf) of \hat{v} is given by

$$pmf(\hat{v}) = \begin{cases} 0.64, & \text{where } \hat{v} = v \\ 0.04, & \text{otherwise} \end{cases} \quad (2.7)$$

Besides uniform perturbation, there are many other perturbation models with different perturbation distribution. For example, (p, γ) -perturbation is proposed to conceal user's interest in a certain advertisement. The user's interest can be interpreted as a binary variable B where $B = 1$ indicates that the user is interested in the advertisement while $B = 0$ indicates that the user is not interested in the advertisement. (p, γ) -perturbation can be expressed by the following formula

$$\hat{B} = \begin{cases} B, & \text{with probability } p, \\ 1, & \text{with probability } (1 - p)\gamma, \\ 0, & \text{with probability } (1 - p)(1 - \gamma) \end{cases} \quad (2.8)$$

Similar to uniform perturbation, the implementation of (p, γ) -perturbation can be considered as tossing two biased coins with head probability of p and γ respectively. Each user can pick p and γ by one of the following two rules:

- *Fixed rule.* Each user can use fixed p and γ to perturb the sensitive values. And the fixed p and γ is known by the adversary.
- *Randomized rule.* Each user picks the values for p and γ from some certain known distributions. The distributions for p and γ are independent with each other and can be different.

The above mentioned perturbation techniques utilize certain kinds of distribution over the domain of sensitive value to perturb the original value with a pre-defined probabilities. It is easy to see that these techniques are suitable for attributes with discrete domain to randomly generate values from probability mass functions. As for numerical values, such as longitude and latitude attributes, additive noise is normally used to perturb the original values.

The general idea of additive noise is to add a random noise δ to the original value v and replace v with the noisy result $\hat{v} = v + \delta$. Typically, the random noise δ is drawn from some distributions independently so that the overall statistical properties, such as means and correlations, are retained. This class of methods gains much popularity in the communities like data mining and participatory sensing, where people care about the overall statistical information rather than individual's record.

A major drawback of additive noise is that it usually needs a large amount of noise to preserve privacy, and the random noise overwhelms the original features contained in the true data. A smart way to cope with this challenge is to project the original data to another space while some features and relationships in the original space are retained. Noise or randomness can be injected into the projection process to preserve privacy so that the original data cannot be recovered by an invert process.

A loss transformation on data can be a projection that converts the original data to a lower dimensional space since in general this type of projection is not reservable. The basic observation is that a high dimensional data point can be uniqueness projected to a lower dimensional data point while it is usually not possible to recover the high dimensional data point only using the lower dimensional data point, since there is information loss in the process of projection. Thus, this type of projection is adopted in [65, 116], and usually follows three steps:

- Choose a dimensionality d , which is lower than the original dimensionality m ;
- Construct a projection matrix \mathbf{A} of size $d \times m$;
- Project each data point $\mathbf{x} \in \mathbb{R}^m$ into $\mathbf{z} \in \mathbb{R}^d$ via the projection $\mathbf{z} = \mathbf{A}^\top \mathbf{x}$.

Even if the projection matrix \mathbf{A} is known by the adversary, it is usually not possible to recover \mathbf{x} from \mathbf{z} . Thus, the privacy information in \mathbf{x} is not visible in \mathbf{z} . The key challenge to enable such technique is the construction of the projection matrix \mathbf{A} so that \mathbf{z} still conveys the same useful information as \mathbf{x} . As such, additional ingredients are put onto the projection matrix to define optimality or usefulness of the results. In [85, 116], the relative distance between data points are kept, i.e., among any three data points $\mathbf{x}_i, \mathbf{x}_j, \mathbf{x}_k$, the relationship that \mathbf{x}_j is more similar to \mathbf{x}_i than \mathbf{x}_k is retained after projection. This type of constraint can be written as

$$\begin{aligned} & \|\mathbf{A}^\top (\mathbf{x}_i - \mathbf{x}_j)\|_2^2 \leq \|\mathbf{A}^\top (\mathbf{x}_i - \mathbf{x}_k)\|_2^2, \\ & \forall i, j, k \in \left\{ i, j, k : \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \leq \|\mathbf{x}_i - \mathbf{x}_k\|_2^2 \right\} \end{aligned} \quad (2.9)$$

However, some important relationships between the feature vectors, such as Euclidean distances and inner products, are lost in the process of dimensionality-reducing transformation. For example, the inner product $\mathbf{x}_i^\top \mathbf{x}_j$ is not identical to $\mathbf{x}_i^\top \mathbf{A} \mathbf{A}^\top \mathbf{x}_j$ unless \mathbf{A} is an orthogonal matrix, which, however, necessarily preserves dimensionality. Therefore, distortion introduced by dimensionality reduction can lead to the performance degradation in using the projected data for certain tasks.

Instead of employing dimensionality-reducing projection, random noises are injected into the projection to preserve privacy. Pickle [65] is a most recent work that applies such technique to participatory sensing. The target of Pickle uses

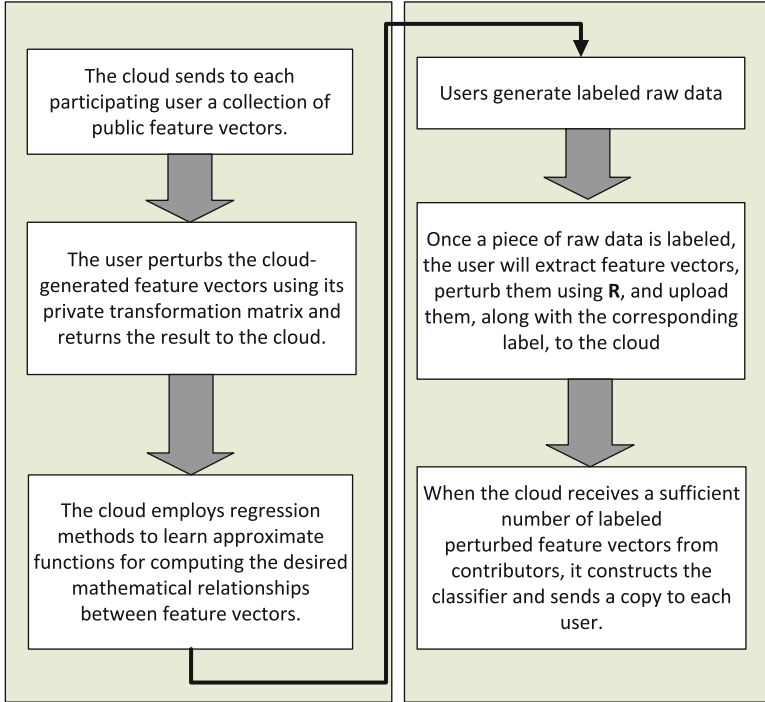


Fig. 2.7 Work flow of Pickle

participatory sensing to build a classifier while keeping each contributor's data private. Pickle first learns a statistical model to compensate the data distortion in the projection. And then Pickle reconstructs the original inner products and distance from the perturbed data, based on which Pickle constructs the final classifier. The work flow of Pickle is illustrated in Fig. 2.7.

The key intuition in Pickle is using regression (the first three blocks in Fig. 2.7) to learn the most important covariance structures in the underlying training data without being able to regenerate the original feature vectors.

In the first step, the cloud randomly generates public feature matrix Z and sends Z to each user. Then, each user perturbs Z in exactly the same way as the user would perturb the actual training feature vectors. To do this, each user keeps a private random matrix R_u to perturb Z into $R_u^T Z$. However, this approach has the following vulnerability. If the private R_u can be recovered as long as Z is invertible, i.e., the cloud can recover R_u by computing $R_u^T Z Z^{-1}$ (note that Z is known by the cloud). This would lead to privacy disclosure. To address this issue, each user adds an additive random noise matrix δ_u to Z . Note that although Z is publicly known, δ_u is kept private to each user. Then, the cloud would recover the original relationships from the perturbed feature vectors. After learning the original relationships, it is ready to construct pattern classifiers using training samples contributed by users.

2.2.1 Privacy Measure

Since random perturbation does not focus on the exact individual or attribute that the adversary can successfully attack by linking its sensitive attribute with the true value, but concentrates on how to camouflage the true values by replacing them with some random variables. The random variables have impacts on the adversaries' beliefs on the true values. The intuitive measure on privacy level is to quantify the information that can be disclosed by how much it would change the adversaries' beliefs on the true values to observe the perturbed values. Generally, this type of privacy measures try to ensure privacy protection by limiting the difference between the prior and posterior beliefs on the true values. In the following part, we will briefly describe some privacy measures for random perturbation.

Uniform perturbation can provide privacy guarantee measured by the difference in the adversary's prior and posterior beliefs. $\rho_1 - \rho_2$ privacy [36] and $\delta - growth$ [94] are two privacy models using such privacy measure. Let $\Pr[v]$ and $\Pr[v|\hat{v}]$ denote the prior and posterior beliefs on the sensitive value v before/after observing \hat{v} . $\rho_1 - \rho_2$ privacy bounds the prior and posterior beliefs by

$$\begin{aligned} \Pr[v] < \rho_1 &\Rightarrow \Pr[v|\hat{v}] < \rho_2, \\ \text{and } \Pr[v] > \rho_2 &\Rightarrow \Pr[v|\hat{v}] > \rho_1, \end{aligned} \quad (2.10)$$

where $0 < \rho_1 < \rho_2 \leq 1$.

Similarly, $\delta - growth$ bounds the prior and posterior beliefs by

$$\Pr[v|\hat{v}] - \Pr[v] < \delta, \quad (2.11)$$

where $0 < \delta < 1$ bounds the growth in the belief on sensitive value v by observing \hat{v} .

As an adaptation of $\rho_1 - \rho_2$ privacy, (d, γ) -privacy is proposed in [84] to bound difference in prior and posterior beliefs, and provides a provable guarantee on privacy as well as utility. Let $\Pr[v]$ and $\Pr[v|\hat{v}]$ denote the prior and posterior beliefs on the sensitive value v before/after observing \hat{v} . The formal definition of (d, γ) -privacy is given as follows.

Definition 4 ((d, γ)-Privacy). Let $\Pr[v]$ and $\Pr[v|\hat{v}]$ denote the prior and posterior beliefs on the sensitive value v before/after observing \hat{v} . A random algorithm is (d, γ) -private if the following conditions hold for all d -independent adversaries

$$\begin{aligned} \frac{d}{\gamma} &\leq \frac{\Pr[v|\hat{v}]}{\Pr[v] \leq d}, \\ \text{and } \Pr[v|\hat{v}] &\leq \gamma. \end{aligned} \quad (2.12)$$

Where adversaries are d -independent if the prior belief satisfies the conditions $\Pr[v] \leq d$ or $\Pr[v] = 1$ (d, γ)-privacy achieves a reasonable trade-off between privacy and utility when the prior belief is small.

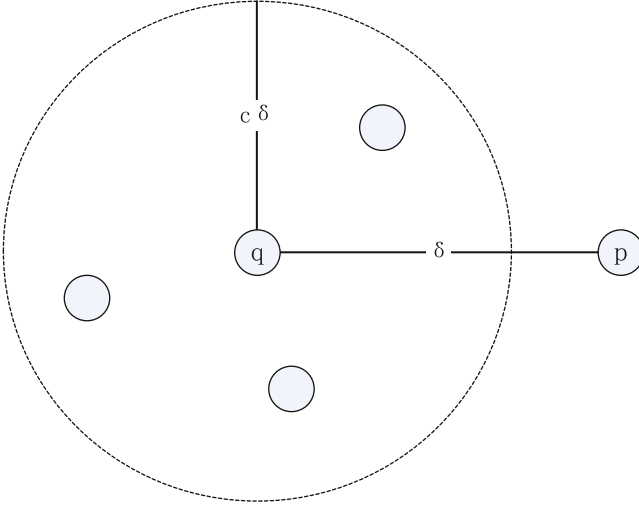


Fig. 2.8 (c, t) -isolation

It is reported in [14] that privacy can be interpreted as the adversary's power of isolating an individual should not be enhanced by observing the released results. Based on this intuition, (c, t) -isolation is proposed. Denote the original data point as p , the corresponding data point inferred by adversary as q , and δ as the distance between p, q . As illustrated in Fig. 2.8, it is said that q (c, t) -isolates p if there are fewer than t data points falling into a ball of radius δ centered at q . This model uses distance to measure privacy level, which is suitable for projection based perturbation.

2.3 Differential Privacy

Differential privacy is the most popular privacy model used in database systems. In this section, we first introduce some basic concepts in the differential privacy model. Then, we interpret differential privacy in terms of priori and posterior beliefs. Finally, we review the state-of-the-arts that are related to location data.

2.3.1 Differential Privacy Model

We first introduce some basic concepts of differential privacy. The intuition of differential privacy is that the removal or addition of a single record does not significantly affect the outcome of any analysis. The following is the formal definition of ϵ -differential privacy in the *non-interactive setting* [12], where ϵ specifies the degree of privacy ensured.

Definition 5 (ϵ -Differential Privacy). A mechanism \mathcal{M} provides ϵ -differential privacy for an SU u if for any possible sets of sensing reports $\mathbf{R} = [\mathbf{r}_1, \dots, \mathbf{r}_u, \dots, \mathbf{r}_U]$ and $\mathbf{R}' = [\mathbf{r}_1, \dots, \mathbf{r}'_u, \dots, \mathbf{r}_U]$ differing only on u 's sensing data,

$$\left| \ln \frac{\Pr[\mathcal{M}(\mathbf{R}) = \mathbf{O}]}{\Pr[\mathcal{M}(\mathbf{R}') = \mathbf{O}]} \right| \leq \epsilon, \quad (2.13)$$

for all $\mathbf{O} \in \text{Range}(\mathcal{M})$, where $\text{Range}(\mathcal{M})$ is the set of possible outputs of \mathcal{M} .

The parameter $\epsilon > 0$ specifies the level of privacy. Specifically, lower value of ϵ ensures stronger privacy. Normally, ϵ is set to be small enough (e.g., 0.1) to make sure that $\Pr[\mathcal{M}(\mathbf{R}) = \mathbf{O}]$ and $\Pr[\mathcal{M}(\mathbf{R}') = \mathbf{O}]$ are roughly the same, meaning that the output \mathbf{O} is insensitive to the change of any single individual's data.

From the viewpoint of an adversary, (2.13) can be rewritten as

$$\left| \ln \frac{\Pr[\mathbf{O}|\mathbf{R}, \mathcal{M}]}{\Pr[\mathbf{O}|\mathbf{R}', \mathcal{M}]} \right| \leq \epsilon. \quad (2.14)$$

We denote $\hat{\mathbf{R}} = [\mathbf{r}_1, \dots, \mathbf{r}_{u-1}, \mathbf{r}_{u+1}, \dots, \mathbf{r}_U]$, and assume that each SU's sensing data is independent of each other [18, 74]. Applying *Bayesian rule* on the LHS of (2.14), we have

$$\left| \ln \frac{\Pr[\mathbf{O}|\mathbf{R}, \mathcal{M}]}{\Pr[\mathbf{O}|\mathbf{R}', \mathcal{M}]} \right| = \left| \ln \frac{\Pr[\mathbf{r}_u|\mathbf{O}, \hat{\mathbf{R}}, \mathcal{M}] \Pr[\mathbf{r}'_u|\hat{\mathbf{R}}, \mathcal{M}]}{\Pr[\mathbf{r}'_u|\mathbf{O}, \hat{\mathbf{R}}, \mathcal{M}] \Pr[\mathbf{r}_u|\hat{\mathbf{R}}, \mathcal{M}]} \right| = \left| \ln \frac{\Pr[\mathbf{r}_u|\mathbf{O}] \Pr[\mathbf{r}'_u]}{\Pr[\mathbf{r}'_u|\mathbf{O}] \Pr[\mathbf{r}_u]} \right|. \quad (2.15)$$

Combining (2.14) and (2.15), we derive

$$e^{-\epsilon} \cdot \frac{\Pr[\mathbf{r}_u]}{\Pr[\mathbf{r}'_u]} \leq \frac{\Pr[\mathbf{r}_u|\mathbf{O}]}{\Pr[\mathbf{r}'_u|\mathbf{O}]} \leq e^{\epsilon} \cdot \frac{\Pr[\mathbf{r}_u]}{\Pr[\mathbf{r}'_u]}. \quad (2.16)$$

$e^{-\epsilon}$ and e^{ϵ} approach 1 as ϵ decreases, which implies that adversaries obtain roughly no extra information about SU's sensing data by observing \mathbf{O} , given the condition that ϵ is small enough.

The standard mechanism to achieve differential privacy utilizes the *sensitivity of a mapping*, which is defined as follows:

Definition 6 (Sensitivity of a Mapping). For any mapping $f : D \rightarrow \mathbb{R}^d$, the sensitivity of f is

$$\Delta f \triangleq \max_{\mathbf{D}, \mathbf{D}'} \| f(\mathbf{D}) - f(\mathbf{D}') \|_1, \quad (2.17)$$

for all input matrices \mathbf{D}, \mathbf{D}' differing at most one user's record.

To ensure that an output $f(\mathbf{D})$ is ϵ -differential private, one standard mechanism [34] is to add random noise to $f(\mathbf{D})$, such that the noise follows a zero-mean Laplace distribution with noise scale of $\frac{\Delta f}{\epsilon}$, denoted as $\text{Lap}(\frac{\Delta f}{\epsilon})$.

One principle mechanism to achieve differential privacy is exponential mechanism [69], which is suitable for algorithms whose outputs are not real numbers or make no sense after adding noise. The exponential mechanism selects an output from the output domain, $o \in \mathcal{O}$, that is close to the optimum with respect to a utility function while preserving differential privacy. The exponential mechanism assigns a real valued utility score to each output $o \in \mathcal{O}$, where outputs of higher scores are assigned with exponentially greater probabilities. Let the sensitivity of the utility function be $\Delta u = \max_{T, T', o} |u(T, o) - u(T', o)|$. The probability associated with each output is proportional to $\exp\left(\frac{\epsilon u(T, o)}{2 \Delta u}\right)$.

Theorem 1 ([69]). *Given a utility function $u : (T \times \mathcal{O}) \rightarrow \mathbb{R}$, a mechanism \mathcal{A} that selects an output o with probability proportional to $\exp\left(\frac{\epsilon u(T, o)}{2 \Delta u}\right)$ satisfies ϵ -differential privacy.*

Another generally used mechanism that fits into differential privacy is Laplace mechanism. Laplace mechanism is suitable for the case where the attribute values are real numbers. The standard way to use Laplace mechanism to achieve differential privacy is to add Laplace noise to the original output of a function. Formally, taking the data set \mathbf{D} , a function $f : D \rightarrow \mathbb{R}^d$, and the differential privacy parameter ϵ as inputs, the additive Laplace noise follows the p.d.f.

$$p.d.f.(x|\lambda) = \frac{1}{2\lambda} e^{|x|/\lambda}, \quad (2.18)$$

where x is the magnitude of the additive noise, λ is a parameter determined by the sensitivity of f as well as differential privacy parameter ϵ .

Theorem 2 ([34]). *Given a function $f : D \rightarrow \mathbb{R}^d$ over an arbitrary input data set D , the mechanism \mathcal{A} that modifies the output of f with the following formula*

$$\mathcal{A}(D) = f(D) + \text{Lap}(\Delta f/\epsilon) \quad (2.19)$$

satisfies ϵ -differential privacy, where $\text{Lap}(\Delta f/\epsilon)$ draws from a laplace distribution with $p.d.f.(x|(\Delta f/\epsilon))$.

To analyze the privacy level in a sequence of operations, the composability of differential privacy is introduced [70], which ensures privacy guarantees for a sequence of differentially private computations. For a sequence of computations that each provides differential privacy in isolation, the privacy parameter values add up, which is referred to as *sequential composition*. The properties of sequential composition is stated as follows.

Theorem 3 ([70]). *Let each computation A_i provides ϵ_i -differential privacy. The sequence of A_i provides $\sum_i \epsilon_i$ -differential privacy.*

In a special case that the computations operate on disjoint subsets of the data set, the overall privacy guarantee depends only on the worst of the guarantees of each computation, not the accumulation. This is known as *parallel composition*.

Theorem 4 ([70]). *Let each computation A_i provides ϵ_i -differential privacy for each D_i , where D_i are disjoint subsets of the original data set. Then, the sequence of A_i provides $(\max(\epsilon_i))$ -differential privacy.*

Recently differential privacy has gained considerable attention as a substitute for anonymization approaches. Numerous approaches are proposed for enforcing ϵ -differential privacy in releasing different types of data. Several data types are related with the data in location privacy. For example, location data and sensing data from a user can be considered as a tuple in histogram data or contingency data table; a location-based query can be considered as a item set in set-valued data. In the following part of this section, we will review several differential privacy approaches on different data types.

2.3.2 Applying Differentially-Private to Set-Valued Data

Publishing different types of data is studied, such as set-valued data [17], decision trees [38], frequent items in transaction data [95], search logs [45, 58], and aggregated results on time series data [83]. These types of data are generally referred to as set-valued data in which each individual is associated with a set of items drawn from a discrete domain. The following table is an example of a set-valued data, where I_i is an item, and the discrete domain of items is $D_I = \{I_1, I_2, I_3, I_4\}$. A taxonomy tree can be used to present all possible generalizations of a item set. A context-free taxonomy tree, which is defined in [17], is a taxonomy tree where the internal nodes the tree are sets of multiple leaves. Note that “context-free” means that the internal nodes in the tree do not have any physical meanings. Figure 2.9 presents a context-free taxonomy tree for items with domain $D_I = \{I_1, I_2, I_3, I_4\}$. According to the taxonomy tree, an item can be generalized to one of its ancestors in the taxonomy tree. We can see that in this example, an item can be generalized to an internal node that contains the item, e.g., I_2 can be generalized to $I_{\{1,2\}}$ but not $I_{\{3,4\}}$.

Intuitively, to release a set-valued data set with differential privacy, a naive method can be directly adding Laplace noise to each possible query on the set-valued data set: first, generating all possible item sets from the item domain, then counting the appearance of each item set in the data set, and adding independent random Laplace noise to each count. However, this simple method has two main drawbacks that make it infeasible to implement: (1) The number of all possible item sets grow exponentially with the number of possible items, so it cannot scale to large data set. (2) the additive noise scale to a new item set accumulates exponentially, which makes the results overwhelmed by noise in a large data set.

Fig. 2.9 A context-free taxonomy tree

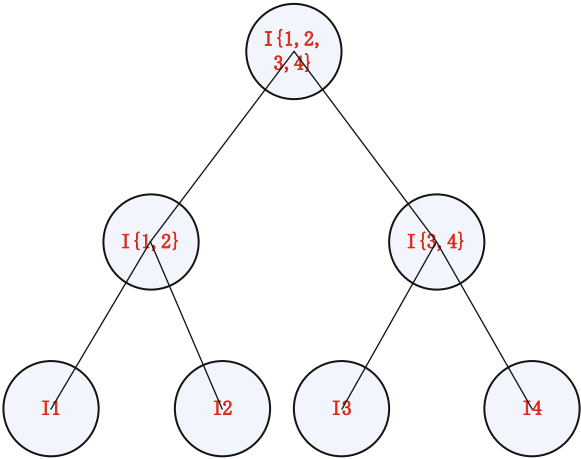


Table 2.3 An example of set-valued data

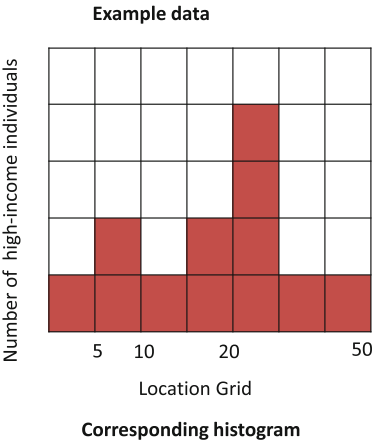
Individual	Items
1	I_1, I_2
2	I_1, I_3
3	I_1, I_2, I_3, I_4
4	I_1
5	I_2, I_4
6	I_3, I_4

To tackle the above two challenges in large-size set-valued data set, a differentially-private sanitization algorithm is proposed in [17] to protect set-valued data publishing by recursively partitioning a set-valued data set based on a context-free taxonomy tree. First, a context-free taxonomy tree is constructed. Then, all records in the data set are generalized to a single partition with the same generalized value. For example, all records in Table 2.3 can be generalized to a partition with generalized value $I_{\{1,2,3,4\}}$, which is the root in the taxonomy tree. Then, a partition is recursively cut into a set of disjoint sub-partitions with more specific generalized values. The partitioning continues recursively in a top-down manner and stops when the partition is considered to be “unpartitionable”, that is, no further partitioning can be applied on the partition. To ensure differential privacy, the “unpartitionability” is determined in a noisy way. Finally, for each unpartitionable partition, the number of records in the partition is counted, and random Laplace noise is added to the count.

To achieve differential privacy, the partitioning procedure cannot be deterministic. Probabilistic operations are required to perform partitioning. Specifically, noise is added in the determination of unpartitionability. Since there are a sequence of operations, a certain portion of privacy budget ϵ is required to obtain the noisy size added to each operation. Thus, privacy budget allocation is required to carefully allocate the total privacy budget ϵ to each probabilistic operation to avoid unexpected algorithm termination. A naive allocation scheme is to bound the

Fig. 2.10 A context-free taxonomy tree

ID	Location Grid	Income
1	17	High
2	6	High
3	7	High
4	11	Low
5	18	High
6	16	High
7	1	High
...



maximum number of partitioning operations and assign an equal portion to each operation. A more sophisticated adaptive scheme is proposed in [17] and it shows better results. The adaptive scheme reserves half portion of ϵ to add noise to counts, and the rest $\epsilon/2$ budget is used to guide partitioning. For each partitioning operation, the maximum number of partitioning operation needed in the future is estimated. And then based on the estimated number, a privacy budget is assigned. The portion of privacy budget assigned to a partitioning operation is further allocated to its sub-partitions determine the unpartitionability. Since all sub-partitions from the same partition operation contain disjoint individuals, the privacy budget portion used on each sub-partition can be the same and doesn't add up according to the parallel composition theory.

2.3.3 Applying Differentially-Private to Histogram Data

A histogram is a representation of tabulated frequencies, which are drawn as adjacent rectangles or bins with an area or height proportional to the frequency of the individuals in the bin. Histogram is usually used in database systems as an effective way to summarize statistical information in numerical domains. Figure 2.10 draws an example of histogram and corresponding data.

Formally, for a certain series of counts $D = \{x_1, x_2, \dots, x_n\}, \forall x_i \in \mathbb{R}^+$, a k -bin histogram merges neighboring counts into k groups, i.e., a k -bin histogram can be presented as $H = \{B_1, B_2, \dots, B_k\}$ where each bin $\{B_i\}$ covers an interval $[l_i, r_i]$ and a count $c_i \in [1, n]$ is associated with B_i . c_i is defined as total number of records falling into the interval $[l_i, r_i]$. In a valid histogram, all bins should cover the whole interval without any overlap between them.

The error for a certain bin B_i is defined by:

$$Error(B_i, D_i) \triangleq |c_i - \sum_{x_j \in [l_i, r_i]} x_j|^2. \quad (2.20)$$

Based on (2.20), the *Sum of Squared Error* (SSE) of histogram H is obtained by:

$$E(H, D) \triangleq \sum_{i=1}^k Error(B_i, D_i).$$

It is obvious that letting $c_i = \frac{\sum_{x_j \in [l_i, r_i]} x_j}{r_i - l_i + 1}$ for all i results in a minimal SSE $E(H, D)$. Histogram construction usually targets at finding the optimal histogram of a count sequence in terms of minimizing the SSE. It has been shown that the optimal histogram construction can be achieved via dynamic programming and costs $O(n^2k)$. The dynamic programming problem can be formulated as

$$E(k, D) = \min_{B_k^i \in D} (E(k-1, D \setminus B_k^i) + Error(B_k^i)), \quad (2.21)$$

where $E(k, D)$ is the minimal SSE for partitioning D into a k -bin histogram, B_k^i the i th sample output for the k th bin B_k , $E(k-1, D \setminus B_k^i)$ the error for constructing partial histogram $H \setminus B_k^i$ with $k-1$ bins.

Histogram publishing via differential privacy is investigated in [12, 49, 111]. Blum et al. [12] divides the input counts into bins with roughly the same counts to construct one-dimensional histogram. And then standard Laplace mechanism is applied on each count independently. According to Sect. 2.3.1, differential privacy is achieved with this simple method. However, this simple method overlooks a fact that the accuracy of a noisy histogram representation depends on its structure: a histogram with larger bins is usually more accurate than a histogram with smaller bins. This is because a histogram with larger bins needs smaller noise scale to satisfy the same level differential privacy.

By observing that the accuracy of a differential privacy compliant histogram depends heavily on its structure, Xu et al. [111] proposes two algorithms with different priorities for information loss and noise scales. The first algorithm consists of two steps: in the first step, Laplace noise is added to each original count x_i independently according to standard Laplace mechanism, where the noisy count sequence can be expressed as $\hat{D} = \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n\}$; in the second step, the optimal histogram based on noisy count sequence \hat{D} is computed via dynamic

Algorithm 1 NoiseFirst Algorithm

Require: Count sequence D ; bin number k ; privacy parameters ϵ , count upper bound A ;

Ensure: Noisy histogram \hat{H} ;

- 1: **for** each x_i in a count sequence $D = \{x_1, x_2, \dots, x_n\}$ **do**
- 2: add independent Laplace noise with magnitude $1/\epsilon$;
- 3: **end for**
- 4: $\hat{D} \leftarrow \{\hat{x}_1, \hat{x}_2, \dots, \hat{x}_n\}$;
- 5: Construct the optimal histogram based on \hat{D} via dynamic programming (2.21);
- 6: Return the histogram $\hat{H} = \{\hat{B}_1, \hat{B}_2, \dots, \hat{B}_k\}$ where $\hat{B}_i = \frac{\sum_{\hat{x}_j \in [l_i, r_i]} \hat{x}_j}{r_i - l_i + 1}$;

Algorithm 2 StructureFirst Algorithm

Require: Count sequence D ; bin number K ; privacy parameters ϵ , count upper bound A ;

Ensure: Noisy histogram \hat{H} ;

- 1: Construct the optimal histogram based on D via dynamic programming (2.21), and keep $Error(B_k^i)$, $E(k, D)$, for all $k \in \{1, \dots, K-1\}$, B_k^i , D ;
- 2: $r_K \leftarrow n$;
- 3: **for** each k from $K-1$ to 1 **do**
- 4: **for** each possible k th bin $B_{k,i}$ **do**
- 5: Compute the error $E(k, D, B_k^i) = E(k-1, D \setminus B_k^i) + Error(B_k^i)$;
- 6: **end for**
- 7: Select $B_k \leftarrow B_k^i$ with probability $\propto \exp\left(-\frac{\epsilon_1 E(k, D, B_k^i)}{2K(2F+1)}\right)$;
- 8: $D \leftarrow D \setminus B_k$;
- 9: Add independent Laplace noise of magnitude $\epsilon_2(r_i - l_i + 1)$ to each count;
- 10: Return the histogram $\hat{H} = \{\hat{B}_1, \hat{B}_2, \dots, \hat{B}_k\}$;
- 11: **end for**

programming (2.21). This algorithm is referred to as NoiseFirst since noise is added before histogram construction. According to standard mechanism [12], the sensitivity of NoiseFirst is 1 and thus Laplace noise with magnitude of $1/\epsilon$ preserves ϵ -differential privacy on constructed histogram. Algorithm 1 summarizes this method.

Another counterpart algorithm is referred to as StructureFirst, where histogram construction comes before adding noise. The motivation for StructureFirst is that NoiseFirst does not leverage the reduced sensitivity after merging neighboring bins. To address this issue, StructureFirst construct histogram on the original data before adding noise to counts. Algorithm 2 summarizes the flow of StructureFirst. First, the optimal histogram is constructed based on the original count sequence D rather than noisy count sequence. The intermediate results in the process of dynamic programming are preserved to guide random bin merging. To add noise to the result histogram, StructureFirst adopts exponential mechanism to randomize the selection

ID	Location Grid	Income
1	17	High
2	6	High
3	7	High
4	11	Low
5	18	High
6	16	High
7	1	High
...

ID	Latitude	Longitude	Income
1	1	8	High
2	3	5	High
3	4	5	High
4	4	2	Low
5	4	4	High
6	6	1	High
7	2	3	High
...

Fig. 2.11 Contingency table

of bin's boundaries, where the probability of selecting a possible bin is proportional to the exponential function of the corresponding SSE $\exp\left(-\frac{\epsilon_1 E(k, D, B_k^l)}{2K(2F+1)}\right)$. The result histogram is proved to be ϵ -differentially-private.

The problem of releasing a set of consistent marginals of a contingency table is studied in [9] and [32], in which correlations among the marginals are extracted to reduce the additive noise. The marginals of contingency consists of multi-dimensional attributes and a sequence of counts. It can be seen as a multi-dimensional version of histogram. For example, the table in the upper side of Fig. 2.11 can be presented as a histogram, as shown in Fig. 2.10, while if we treat Income as attribute and count number of records falling into a two-dimensional (i.e., Location Grid and Income) region, it becomes marginals of the contingency table. The marginals can be high dimensional. As illustrated in Fig. 2.10, the Location Grid can be further de-composed as longitude and latitude and thus the resulting marginals becomes three-dimensional.

Count query on multiple dimensional tables can be interpreted as multiple cells. Each cell aggregates a subset of the rows in the table with corresponding counts, as shown in Fig. 2.12. The cells can be regarded as reflections of the original table

Fig. 2.12 Cell tables

Latitude	Income	Count
1-3	High	3
4-6	High	3
1-3	Low	0
4-6	Low	1

Cell {Latitude, Income}

Longitude	Income	Count
1-4	High	3
5-8	High	3
1-4	Low	1
5-8	Low	0

Cell {Longitude, Income}

Latitude	Longitude	Income	Count
1-3	1-4	High	1
1-3	1-4	Low	0
1-3	5-8	High	2
1-3	5-8	Low	0
4-6	1-4	High	2
4-6	1-4	Low	1
4-6	5-8	High	1
4-6	5-8	Low	0

Cell {Latitude, Longitude, Income}

on different subsets of dimensions with aggregated measures. However, a adversary can still infer sensitive information about an individual by just look at difference cells. To address this issue, instead of publishing exact counts, many proposals adopt differential privacy model to add random noise to the counts.

A simple way to release cell tables with differential privacy is directly applying Laplace mechanism. The first approach is to independently add Laplace noises with proper scales to each cell. The limitation of this approach lies in the fact that for a d -dimensional table there are 2^d cell tables in total. To satisfy ϵ -differential privacy, each cell needs to add Laplace noise $Lap(2^d/\epsilon)$, which grow exponentially with table dimensions. This significant amount of noise in a high dimensional table renders cell tables useless. Another issue is consistency among different cells. A key observation is that there exists correlations between cells, that is, lower-dimensional cells can be derived by merging some rows in higher-dimensional cells. If noises are added to each cell independently, the results may not be consistent across different dimensional cells. The inconsistency can be leveraged by an adversary to infer user's

private data. For example, as all noises are independent, an adversary can compute the mean of multiple inconsistent counts to derive a more accurate results. Another issue of inconsistency is that data analyzers may interpret the inconsistencies as an evidence of bad data. A user study conducted by Microsoft [72] reported that only small inconsistencies in modified data were acceptable.

To address this issue, Ding et al. [32] divides the set of cell tables into two subsets. A subset of cell tables are selected and computed directly from the original data table. Standard Laplace noises are injected to these cell tables. This subset is called the initial subset. Then, based on the noisy initial subset, the count measure for the rest cell tables are computed. When the initial subset is larger, each of its cell tables requires more noises to preserve the same level of differential privacy, while on the other hand, the rest cell tables whose count measure is computed from the initial subset contain less noise. To choose a proper tradeoff that minimizes overall noise, a noise control framework is proposed in [32], in which two scenarios under the policy of Ministry of Health are discussed. In the first scenario, a set of cell tables are required to be released, and the objective is to minimize the maximum noise in the cell tables. The other scenario is the importance of each cell table is indicated by a weight function. The question is to decide which cell tables to release so that the noise level in each cell table is bounded by a pre-defined threshold. The objective is to maximize the sum of the weights of all released precise cell tables.

To achieve these goals, two optimization problems are formulated to select the initial subset in these two scenarios, respectively. These two optimization problems are proved to be NP-hard. The proof uses a non-trivial reduction from the *Vertex Cover problem* in degree-3 graph. A vertex cover of a graph G is a set S of vertices such that each edge of G is incident to at least one vertex in S . The Vertex Cover problem is to find the smallest vertex cover of a given graph. A brute-force approach for these problems is to enumerate all possible choices of the initial subset, which takes $O(2^d)$ time where d is the dimension of the original table. The brute-force approach is not practical for high-dimensional table. Approximation algorithms with polynomial time complexity are proposed: Bound Max Variance and Publish Most.

To cope with the first scenario, whose target is to minimize the maximum noise in cell tables, we can find the minimum noise threshold η such that there exists feasible solutions. Specifically, suppose we have algorithm for a subproblem $SubProb(S_i, n, \eta)$, which is described as: for a given threshold η and a positive integer n , is there an initial subset S_i with n cell tables that for all cell tables in S_i , the required noise is no larger than η ? If the answer is yes, we set $SubProb(S_i, n, \eta) = YES$, otherwise $SubProb(S_i, n, \eta) = NO$. Then the problem can be interpreted as finding the minimum η such that $SubProb(S_i, n, \eta) = YES$. To find such η efficiently, binary search instead of brute-force search is used. The details are provided in Algorithm 3 and Algorithm 4. It is proved in [32] that the Bound Max Variance algorithm provides a logarithmic approximation and runs in polynomial time.

Consider the second scenario in which the objective is to maximize the sum of the weights of all released precise cell tables. Suppose the optimal solution for a

Algorithm 3 $SubProb(S_i, n, \eta)$

```

1: compute coverage  $coverage(C)$  for each cell  $C$ ;
2:  $R \leftarrow \emptyset$ ;
3:  $COV \leftarrow \emptyset$ ;
4: repeat the following steps  $n$  times:
5: pick the cell  $C'$  with max  $|coverage(C') - COV|$ ;
6: add  $C'$  to  $R$ ;
7: add  $coverage(C')$  to  $COV$ ;
8: if  $COV$  covers all cell tables then
9:    $S_i = R$ ;
10:  return YES;
11: else
12:  return NO;
13: end if

```

Algorithm 4 Bound Max Variance

```

1:  $\eta_L \leftarrow 0$ ;
2:  $\eta_R \leftarrow 2^{2d+1}/\epsilon^2$ ;
3: while  $|\eta_R - \eta_L| > 1/\epsilon^2$  do
4:    $\eta \leftarrow \frac{\eta_R + \eta_L}{2}$ ;
5:    $\eta_R \leftarrow \eta$ ;
6:   for all  $n = 1, 2, \dots, 2^d$  do
7:     if there exists an  $n$  such that  $SubProb(S_i, n, \eta) = YES$  then
8:        $\eta_L \leftarrow \eta$ ;
9:     end if
10:  end for
11:  return  $\eta_R$ ;
12: end while

```

noise threshold η_o contains a set of n cell tables. It is proved that the problem is equivalent to finding a set of n cells subjected to the weighted sum of their covered cells is as high as possible. A greedy algorithm is considered to solve the problem for each choice of n . As described in Algorithm 5, each iteration finds the cell table that maximize the total weight of the cells who are not previously covered. This cell table and the corresponding newly covered cell tables are added to R and COV , respectively. At the end of the 2^d th iteration, the algorithm selects the best R over all possible n as the initial subset. It has been proven that Algorithm 5 provides $(1 - 1/e)$ approximation with running polynomial in 2^d .

There are several other studies on using differential privacy to release different types of data. As these works are tightly related to location data, this book does not elaborate them. Readers who are interested can refer to the follower papers. The brief ideas of the other state-of-the-arts are described as follows. *Privelet* [110] is proposed to lower the magnitude of additive noise to publish multi-dimensional

Algorithm 5 Publish Most

```

1: for  $n = 1, 2, \dots, |2^d|$  do
2:   for each cell  $C$  do
3:     compute coverage  $\text{coverage}(C)$  based on  $\eta_o, n$ ;
4:   end for
5:    $R \leftarrow \emptyset$ ;
6:    $COV \leftarrow \emptyset$ ;
7:   repeat the following steps  $n$  times:
8:     select a cell table  $C'$  with maximal  $\text{coverage}(C') - COV$ ;
9:     add  $C'$  to  $R$ ;
10:    add  $\text{coverage}(C')$  to  $COV$ ;
11:   end for
12: return the best  $R_s$ 

```

frequency matrix. *DiffGen* [75] connects generalization technique with differential privacy to releases an sanitized data table for classification analysis. DiffGen randomizes the generalization operation and adds noise to count in each partition. Recently, several works [30, 32, 80] try to handle multi-dimension issue in differential privacy. Peng et al. [80] and Cormode et al. [30] introduce differential private indices to reduce errors on multi-dimensional data sets.

2.4 Private Information Retrieval

The aim of private information retrieval (PIR) is to enable a user to retrieve an item from a database without revealing which item is retrieved. Typically, PIR is used in database query, which is related to the database-driven CRNs in which registered SUs need to query available channels from a database. A simple way to achieve this is for the user to query the whole database. Apparently, it is very inefficient and requires large amount of overhead. A natural question arises: can we use less communication overhead while still ensuring the privacy of the retrieval? PIR replies this question with a positive answer. A tutorial on PIR can be found in [114].

Let's describe the problem in a more concrete manner. A database can be modeled as [20]

1. a vector of bits $\mathbf{x} = [x_1, x_2, \dots, x_n]$,
2. and a computational agent can do computations over \mathbf{x} based on a query q .

Based on this database model, the problem of PIR can be stated as:

1. a user retrieves x_i ,
2. i is unknown to the database.

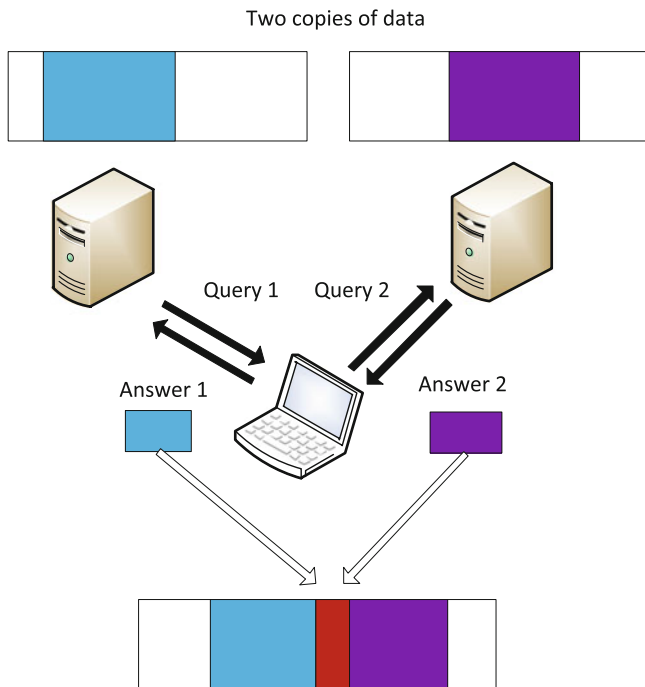


Fig. 2.13 PIR: two databases illustration

Actually, the problem can be more strict: the database cannot know that $i \neq j$. This can be achieved by querying total n bits. Then the question here becomes can the user retrieve x_i with complete privacy while using less bits of communication? It is usually assumed that the user's PIR algorithm is known to the database. In such case, the answers are divided. Here we only outline the major PIR techniques. A detailed survey on PIR can be found in [41, 78]. In [41], the authors conclude several cases:

- If the user adopts a deterministic algorithm, n bits communication is required.
- If the copy of the data is stored only in one database, and the database's computational power is unlimited, then n bits communication is required.
- If the copy of the data is stored only in multiple non-communicating databases, and the database's computational power is unlimited, then less than n bits communication is possible.

Figure 2.13 illustrates a two databases case. The user queries two databases separately, each of which stores a copy of the identical data. Each individual query carries no information about what the user wants to retrieve. The retrieval data is obtained by combining the two answers. The most efficient two-database PIR protocols known today require communication overhead of $\mathcal{O}(n^{1/3})$ [20]. Moreover, PIR techniques for three or more databases have been widely

developed [7, 10, 35, 112]. The lower bounds and upper bounds have been discussed in many research works, but closing the gap between upper and lower bounds for PIR communication complexity is still an open problem.

The core technique adopted to design PIR is error-correcting codes, which are initially designed for reliable transmission over noisy channel, or reliable storage on partially corrupted medium. A category of error-correcting codes called Locally decodable codes or LDCs are leveraged to enable PIR. The core functionality of LDCs is to provide efficient random-access retrieval and high noise resilience simultaneously by reliably reconstructing an arbitrary bit of the message based on a small number of randomly chosen codeword bits.

A (k, δ, ϵ) -LDC encodes an n -bit message \mathbf{x} to an N -bit codewords $\mathbf{C}(\mathbf{x})$, where for each $1 \leq i \leq n$, the i th bit x_i can be recovered with probability $1 - \epsilon$ by a randomized decoding procedure reading k codeword bits, even after $\mathbf{C}(\mathbf{x})$ is corrupted in at most δN bits. PIR and LDCs are strongly related. Short LDCs yield efficient PIR schemes and vice versa. In the following part, we describe how to derive a k -database PIR scheme from any perfectly smooth k -query LDC.

- First, each of the k databases D_1, \dots, D_k encodes a n -bit data \mathbf{x} with a perfectly smooth LDC $\mathbf{C}(\cdot)$.
- Without loss of generality, we assume that a user wants to retrieve the i th bit x_i . Then, the user tosses random coins to generate k random queries q_1, \dots, q_k , such that x_i can be computed from these k queries.
- The user sends k queries q_1, \dots, q_k to corresponding databases D_1, \dots, D_k .
- Each database $D_j, j \in \{1, \dots, k\}$ answers the query q_j with one bit $\mathbf{C}(\mathbf{x})_{q_j}$.
- The user reconstruct x_i by combining $\mathbf{C}(\mathbf{x})_{q_1}, \dots, \mathbf{C}(\mathbf{x})_{q_k}$.

The above protocol preserves the privacy of x_i in that each query q_j is uniformly distributed over the set of codeword coordinates and thus the database D_j learns nothing from q_j . The total communication cost is $k(\log N + 1)$. As for surveys on information-theoretic PIR, you can refer to [97, 113].



<http://www.springer.com/978-3-319-01942-0>

Location Privacy Preservation in Cognitive Radio
Networks

Wang, W.; Zhang, Q.

2014, IX, 76 p. 31 illus., Softcover

ISBN: 978-3-319-01942-0