A New View of Privacy in Social Networks:
Strengthening Privacy during Propagation

Wei Chang
Temple University, USA

Jie Wu
Temple University, USA

ABSTRACT

Many smartphone-based applications need microdata, but publishing a microdata table may leak respondents' privacy. Conventional researches on privacy-preserving data publishing focus on providing identical privacy protection to all data requesters. Considering that, instead of trapping in a small coterie, information usually propagates from friend to friend. The authors study the privacy-preserving data publishing problem on a mobile social network. Along a propagation path, a series of tables will be locally created at each participant, and the tables' privacy-levels should be gradually enhanced. However, the tradeoff between these tables' overall utility and their individual privacy requirements are not trivial: any inappropriate sanitization operation under a lower privacy requirement may cause dramatic utility loss on the subsequent tables. For solving the problem, the authors propose an approximation algorithm by previewing the future privacy requirements. Extensive results show that this approach successfully increases the overall data utility, and meet the strengthening privacy requirements.

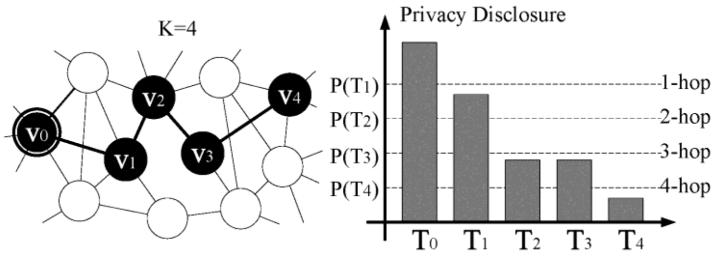
Keywords: Data Utility, Distributed System, Increasing Privacy Levels, Microdata, Multi-Hop Relays, Privacy-Preserving Data Publishing, Sanitization, Social Networks.

INTRODUCTION

Learning others' social features can significantly improve the performance of many mobile social network-related tasks, such as data routing [1], personalized recommendation [2] and social relationship prediction [3]. In these scenarios, a participant needs access to a large volume of personal information in order to spot the pattern [4]. A dataset, which consists of the information at the level of individual respondents, is known as microdata dataset. In order to protect the privacy of each individual respondent, data holders must carefully sanitize (also known as anonymize) the dataset before publishing. In the past decade, many privacy standards have been proposed, such as k-anonymity [5], l-diversity [6], and t-closeness [7].

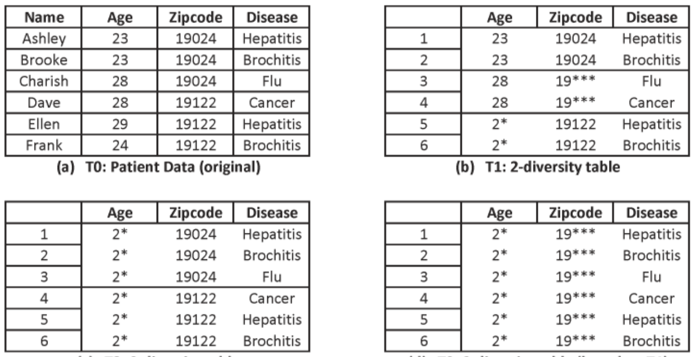
Unlike the conventional centralized database system, where data requesters *directly* interact with data owners, information on a mobile social network is disseminated from user to user via *multi-hop relays*. Considering the well-known limitations with centralized systems, such as system bottlenecks or a single point of attacks problem, in this paper, we study the problem of multi-hop relay-based privacy-preserving data publishing, where a microdata table is gradually propagated from its original owner to distant people. However, under this scheme, the recipients will present different trust-levels regarding to the original data owner. Intuitively, after each time of relay, one should further provide more privacy protections on the data. For example, in Fig. 1(a), along a social path with length K, each user eventually will get one copy of v_0 's table, and we need the tables' privacy to be gradually reinforced, as shown by Fig. 1(b). Data privacy and data utility are naturally at odds with each other [4]: The more privacy a dataset preserves, the less utility the dataset has. This propagation scheme creates a unique problem: `for a group of friends, how can they create a series of tables with maximal overall data utility, and assure that the tables' privacy are increasingly protected at the same time?' To our best knowledge, this unique problem has never been proposed or solved.

Take Fig. 2 as an example. Suppose that l -diversity is the privacy requirement, and the total target propagation distance l is equal to 2. Assume that the corresponding participants are v_0, v_1 and v_2. With the growing of the propagation distance, the parameter l becomes larger and larger (i.e. after the first hop, the table should satisfy 2 -diversity, and after the second hop, it should satisfy 3-diversity). The original dataset is given by Fig. 2(a). Figs. 2(b) and (c) give the results by directly using anonymizing operations on the original table T_0. We can see that the sanitized values are different in these two tables. However, during multi-hop relays, a participant can only observe the table passed from the previous one, and therefore, if v_0 gives T_1 to v_1, v_2 can only obtain Fig. 2(c), instead of T_2. Consider that the tables, which satisfy (l +1)-diversity, must satisfy l -diversity. For v_0 , he has two options for sending the dataset to v_1: he either sends T_3 or T_2. For the first case, the user v_1 only needs to forward T_3 to v_2 without any changes, while for the other case, v_1 should further sanitize T_2 and send the result T_3 to v_2. Clearly, we cannot simply claim which approach is better; this is because it depends on the utility value of each attribute. For instance, if we define that any suppression operation costs 1 unit of utility, then the first option loses a total of 12 units of utility (as 6 units of utility are lost for T_1), while the second one costs 16 units (as 4 utility units are lost for T_2 and 12 for T_3). However, if the age attribute is more important than the zip code, assuming that suppression on age costs 2.5 units of utility and suppression on zip code costs 1, the utility loss of the second option becomes 28, while that of the first option is 30.



(a) a social chain (b) increasing protections

Figure 1 An example of the target problem. In fig. (a), the black nodes consists of a social chain with length 4. The source v0 possesses a data table and wants to propagate it to the other four nodes (from v1 to v4). Since the source node has a different trust level to these destination nodes, the privacy protection of the table's content should be further enhanced after each time of relays. In fig. (b), the dash lines represent the enhanced privacy requirements and the bars stand for the real privacy value of the table.



Name	Age	Zipcode	Disease
Ashley	23	19024	Hepatitis
Brooke	23	19024	Brochitis
Charish	28	19024	Flu
Dave	28	19122	Cancer
Ellen	29	19122	Hepatitis
Frank	24	19122	Brochitis

(a) T0: Patient Data (original)

	Age	Zipcode	Disease
1	23	19024	Hepatitis
2	23	19024	Brochitis
3	28	19***	Flu
4	28	19***	Cancer
5	2*	19122	Hepatitis
6	2*	19122	Brochitis

(b) T1: 2-diversity table

	Age	Zipcode	Disease
1	2*	19024	Hepatitis
2	2*	19024	Brochitis
3	2*	19024	Flu
4	2*	19122	Cancer
5	2*	19122	Hepatitis
6	2*	19122	Brochitis

(c) T2: 3-diversity table

	Age	Zipcode	Disease
1	2*	19***	Hepatitis
2	2*	19***	Brochitis
3	2*	19***	Flu
4	2*	19***	Cancer
5	2*	19***	Hepatitis
6	2*	19***	Brochitis

(d) T3: 3-diversity table (based on T1)

Figure 2 Generated tables based on requirements with different privacy levels.

In order to solve the problem, we first propose two greedy schemes: a bottom-up algorithm and a top-down algorithm. The bottom-up algorithm works at the tuple-level, and it always merges similar tuples into one equivalent class until all classes meet the corresponding privacy requirement. The top-down approach is based on the observation that the larger an equivalent class is, the safer its members are. This approach considers how to split a table into the maximum number of sub-tables, such that each small table has the required privacy protection. By comparing the results of both algorithms, we propose another approximation algorithm for optimally creating a series of tables. Consider that if a table satisfies a higher privacy requirement, the n it also meets the lower one. At each participant, the algorithm does not simply maximize the data utility at its required privacy-level; instead, it takes future, more stringent privacy requirements into consideration. Extensive simulation results show that more data utilities are preserved by adopting our scheme.

The contribution of this chapter is threefold. First, to our best knowledge, we are the first to consider the multiple-level privacy-preserving data publishing problem within a distributed system. Second, since there are two objects with the problem, we propose a bottom-up approach and a top-down one, which give different priorities to the objects. In order to achieve maximal overall utility, we introduce the concept of `forward looking': the sanitization at an earlier phase should not significantly jeopardize the data utilities on the later phase. Lastly, our real data-based evaluation demonstrates that the forward looking scheme can produce a 5.6% higher total utility than the top-down and bottom-up approaches.

RELATED WORK

Privacy-preserving data publishing has received considerable attention in the past decade. Based on the different goals of privacy-preservation and data features, a variety of privacy protection techniques have been designed. Statistical databases are used for the purpose of statistical analysis, and it aims to provide aggregated knowledge. Since intelligent users may use a combination of aggregate queries to derive information about an individual record, statistical databases need privacy protection. Commonly used privacy-preservation techniques in this field include adding random noise while maintaining certain statistical features [8], [9], and making perturbation on the query-results [10], [11]. Differential privacy [12], [13] aims to provide an indistinguishability regarding the existence of an individual respondent, during interactive queries. Unlike traditional approaches, differential privacy puts restrictions on the release mechanism instead of the original dataset.

Unlike a statistical database, a microdata database publishes information at the level of individual respondents [14], and its records are unperturbed. In order to preserve individual respondents' privacy, many privacy metrics have been developed, such as k-anonymity [5], l-diversity [6], and t-closeness [7]. In order to achieve these metrics, certain anonymizing operations are adopted; generalization and suppression [15], [16] to quasi-identifiers are the most two commonly used operations. Essentially, the anonymizing operations divide tuples of a table into multiple disjoint groups, and the tuples from the same group form an equivalence class. k-anonymity requires that the size of each group must be at least k; l-diversity needs each group to contain at least l unique sensitive values, and t-closeness requests the distribution of the sensitive attributes within each group to be similar to the distribution within the original dataset. However, almost all of the existing studies focus on providing a single privacy-preserving level, while, in practice, the recipients are not equally trustable [17], especially on a social network. To our best knowledge, only papers [17], [18] consider multi-level privacy preserving on statistical databases. Due to the differences about the goal of privacy-preservation and anonymizing operations, their approaches cannot be applied on microdata tables. Moreover, they use a centralized scheme, but our problem considers a distributed scenario.

The cost of privacy is the loss of data utility. For a centralized database system, before publishing a table, the data holder first defines a privacy requirement, and then, anonymizes the table with the minimal utility damage. Certain works [4], [19], [20] have been done on this kind of tradeoff between privacy and utility. However, few people have worked on the privacy-utility tradeoff about *multiple related tables*. Unlike the traditional database, propagating a dataset along a social path with enhancing privacy-preserving requirements essentially creates multiple mutually-related tables. In practice, it is often necessary to maximize the overall utilities of the whole tables, instead of simply optimizing one table's utility. In addition, the optimization of overall utility is usually not equivalent to the simple combination of individual tables' optimal results, as we have shown in Section I of the paper.

PROBLEM MODEL AND FORMULATION

Conventional privacy-preserving data publishing problems focus on providing uniform privacy protection to all requesters, and therefore, in their models, the database is the only source of information. However, in real life, information usually propagates from person to person in a distributed way, instead of being trapped in a small coterie. In this paper, we propose a distributed privacy-preserving scheme, which provides a multi-level privacy for different social users. The most significant feature of the proposing scheme is that, after each time of dissemination, the data's privacy is further strengthened. In this section, we present the problem formulation, system model, and some commonly used symbols.

A. Basic Model

On a social network, there is a social chain consisting of $l+1$ participants (also called nodes), assuming$v_0, v_1, v_2, \dots, v_l$. Let $T = \{t_1, t_2, \dots, t_m\}$ be a microdata table of m tuples and n attributes $A = \{A_i\}$. Usually, we have $m \gg n$. In practice, a microdata table may contain several sensitive attributes, such as salary and disease. Since we can regard each possible combination of the sensitive attributes as a data point (vector) in a hyper-dimension, here, we assume that there is only one sensitive attribute, which is A_n. In other words, attributes $\{A_1, A_2, \dots, A_{n-1}\}$ are the quasi-identifiers (non-sensitive attributes), and A_n is the sensitive attribute.

Definition 1: In a table T , tuples with the same quasi-identifiers' values form an equivalent class $< t >$.

For example, in Fig. 2(b), $< t_3 >$ means the equivalent class of the third tuple, and therefore, $< t_3 > =< t_4 >$ = {(28, 19***, Flu); (28, 19***, Cancer)}. Also, in Fig. 2(c), $< t_4 > =< t_5 > =< t_6 >$ = {(2*, 19122, Cancer); (2*, 19122, Hepatitis); (2*, 19122, Bronchitis)}.

Let σ be the selection operation from a table, and $\sigma_i(T_j)$ stands for the ith tuple of table T_j. For instance, in Fig. 2(b), $\sigma_3(T_1)$ means selecting the third tuple from T_1, and therefore, $\sigma_3(T_1)$ ={(28, 19***, Flu)}. In Fig. 2(c), $\sigma_3(T_2)$ ={(2*, 19122, Cancer)}. We use π as the projection operation on a table. $\pi_i(T)$ gives the ith column of table T . For instance, in Fig. 2(a), $\pi_3(T_0)$ ={Hepatitis; Bronchitis; Flu; Cancer; Hepatitis; Bronchitis }. Since we assume that the nth attribute is the sensitive one, $\pi_n(T)$ means the sensitive values in T. For the ease of description, for the rest of the paper, let $\pi(T)$ be the short of $\pi_n(T)$. So, $\pi(T_0) = \pi_3(T_0)$. In addition, any value in table T_k, attribute (column) A_j , tuple (row) t_i is represented as$\sigma_i(\pi_j(T_k))$. In Fig. 2(b), $\sigma_3(\pi(T_1))$ ={Flu}, and $\sigma_{<3>}(\pi(T_1))$. ={Flu, Cancer}.

Along the social chain, v_0 is the original data owner, and based on trust relations, user v_i (i ∈ [1, l]) receives an anonymized table T_i from his previous user, and sends a modified version, T_{i+1}, to the next one. For instance, v_0 sends table T_1 to v_1; based on T_1, the receiver, v_1, creates table T_2 and sends it to v_2, and so on. Essentially, each user locally builds a table based on previous user's data, and therefore, T_i is a sub-table of T_{i-1}. Considering the fact that trust fades along a social path, instead of having l tables with the same privacy-preserving strength, our system builds a series of tables with *increasing* privacy protections. Therefore, table T_{i+1} is the sanitization result of table T_i. For benefiting the whole society on a social chain, the data owner wants to provide the maximal *overall* utility of these tables.

Definition 2: Given a pair of tables T and T', if the value of any tuple in T is covered by the value of T''s corresponding tuple, ∀i ∈ [1, l], ∀j ∈ [1, m], $\sigma_i(\pi_j(T)) \subseteq \sigma_i(\pi_j(T'))$, then we call T' a *sanitization result* of T .

For example, value "19**2" covers "191*2", since the value's range of "191*2" is a subset of the former one. Note that any value covers itself, e.g. "19**2" covers "19**2".

In practice, different attributes may have different levels of importance. For example, for the data published by Screen Actors Guild, the public cares more on the actor/actress's age or weight than their academic performance. In our model, we classify users into several types, and for the tables owned by each type of user, there is a set of utility weights W = $\{w_1, w_2, \dots, w_{n-1}\}$ associated with quasi-identifiers (w_{i} ∈ [0, 1], $\sum_i w_i$ = 1). The larger the weight is, the more important the attribute is. Utility weight is a source-based concept: along a table's propagation path, all participants use the same utility weights since the tables have the same source.

TABLE I. NOTATIONS USED IN THIS PAPER

Notation	Description
δ	a pre-defined privacy threshold, $\delta \in (0, 1)$.
l	the length of propagation.
m	the total number of tuples.
n	the number of attributes.
v_i	the ith node on a social chain, $i \in [0, l]$.
T_i	the database table possessed by v_i.
t_i	the jth tuple (row) in a table, $i \in [0, m]$.
$\langle t \rangle$	an equivalent class of tuple t.
A_i	the quasi-identifiers, $i \in [1, n-1]$.
A_n	the sensitive attribute.
$\sigma_i(T_j)$	select tuple t_i from table T_j.
$\pi_i(T_j)$	project the quasi-identifier value A_i in T_j.
w_i	utility weights for quasi-identifiers.
$p(\cdot)$	the distribution of a given set of values.
$P(T)$	table T's privacy disclosure function.
$U(T)$	table T's data utility function.
$JS(\cdot,\cdot)$	JensenShannon divergence function.
$d(\cdot,\cdot)$	sematic distance between two values.
$\lVert v_i - v_0 \rVert$	social distance from v_i to v_0.
$H(A_n \mid T)$	information gain about sensitive value A_n given T.

B. Problem Formulation

1) Privacy Disclosure Measurement: Let $p(\cdot)$denote the distribution of a given set of values, then $p\big(\pi(T)\big)$ represents the sensitive values' distribution over the whole table , and $p\left(\sigma_{<t>}\big(\pi(T)\big)\right)$ indicates that distribution within an equivalent class$< t >$. For a given table , its privacy disclosure $P(T)$ is measured as the maximum amount of privacy disclosure from its equivalent classes:

$$P(T) = max_{\forall <t> \subseteq T} JS\left(p(\pi(T)), p(\pi_{<t>}(T))\right)$$

where $JS(\cdot,\cdot)$ is Jensen-Shannon divergence function [21], which measures the similarity between two distributions. $JS \in [0, 1]$. The smaller the JS is, the closer the distributions are.

2) Data Utility Measurement: Assume that $\sigma_i(\pi_j(T))$ and $\sigma_i(\pi_j(T'))$ are the same data point appearing in two tables. Due to the sanitization operations, the values of them may not be same. Let $d(\sigma_i(\pi_j(T)), \sigma_i(\pi_j(T')))$ be the sematic distance between them. For instance, by using suppression operations, the distance between "19*22" and "19***" is 2 units. If we use generalization, the distance between age "29" and range "[2 0-30)" is defined as the distance from 29 to the mean age within that age group. For a given attribute, if its values are extremely similar to each other, basically, the whole attribute becomes useless for most of the data mining task. As a result, the utility of an attribute mainly comes from its diversity. If we use sanitization operations to merge all tuples of table T into one equivalent class, which has the same effects as only publishing sensitive attributes, then the resulting table T reaches maximum privacy but minimum utility. For the rest of the paper, we use $\tilde{T}$ to represent the resulting table by sanitizing a given table T into one equivalent class. Let $U(T)$ stand for the data utility of any table , then we have:

$$U(T) = \sum_{i=1}^{n-1}\left[w_i \sum_{j=1}^{m} d(\sigma_j(\pi_i(T)), \sigma_j(\pi_i(\tilde{T})))\right]$$

3) Formulation and Features: By hiding the exact value of attributes, a table can gain more privacy but might lose unity. Along a table's propagation path, the privacy protection on the table should be gradually enhanced, since the trust relations between the original data owner and each receiver is fading. As a result, before disclosing a table to the next recipient, each relay node must further anonymize the table until it reaches a certain privacy level. Here is the problem formulation: for a given table T_0, along a social path with length l, we want to create l tables; each table T_i is the sanitization result of T_{i-1} (i ∈ [1, l]), and these tables satisfy the following requirements:

$$\text{Maximize} \qquad \sum_{i=1}^{l} U(T_i)$$

$$\text{Subject to} \qquad P(T_i) < (l - i + 1)\delta$$

Theorem 1: For a given microdata table T_0 and a pair of source-selected parameters δ, l, there is at least one series of tables
$T_1, T_2, \dots, T_l$ satisfied the increasing privacy requirements $P(T_i) < (l - i + 1)\delta$ for ∀i ∈ [1, l].

Proof: One can delete all attributes of T_0 except the sensitive ones, and let the result $\tilde{T}$ be T_i . On table T_i, all tuples form only one equivalent class, and therefore $P(T) = max_{\forall <t> \subseteq T} JS\left(p(\pi(T)), p(\pi_{<t>}(T))\right) = 0$. Since $(l - i + 1)\delta > 0$ for ∀i ∈ [1, l], the created tables, $T_1, T_2, \dots, T_l$, always satisfy the privacy requirements.

■

Theorem 2: Along the tables' propagation path, the table utility U is non-increasing: $U(T_i) \geq U(T_j)$, for ∀i, j, 1 ≤ i < j ≤ l.

Proof: Because sanitization operations (i.e. generalization and suppression) are irreversible, for ∀i, j, 1 ≤ i < j ≤l, table T_j must be a child-table of table T_i, and therefore, its utility should be less than T_i. Even if there exists a pair of tables T'_i and T'_j, whose table utility satisfies $U(T'_i) < U(T'_j)$, one can simply replace T'_i with T'_j because $P(T'_j) < (l - j + 1)\delta < (l - i + 1)\delta$.

Theorem 3: The target problem: "given table T_0, propagation length l, and a threshold δ, maximize $\sum_{i=1}^{l} U(T_i)$ subject to $P(T_i) < (l - i + 1)\delta$ for ∀i ∈ [1, l]" is NP-hard.

Proof: We want to reduce our original problem to the classic optimal L-diversity problem, which is NP-hard [22]. Consider the simplest case of our problem, where l = 1. Under this special case, our problem is reduced to: given table

$T_0$and a threshold $\delta > 0$, Maximize $U(T_1)$ subject to $P(T_1) < \delta$, where T_1 is a sub-table of T_0. Since the amount of privacy disclosure is measured by Jensen-Shannon divergence, each type of sensitive value must appear at least once within each equivalent class. Assume that there are a total of L types of sensitive values. After we further simplify the problem into the situation that all sensitive values follow a uniform distribution, the problem becomes a typical L-diversity problem. As a result, our target problem is NP-hard. ■

As Theorem. 3 shows, the target problem is NP-hard even if a table is propagated only one hop. Actually, maximizing the overall utility of l tables is more challenging than the one-hop case. Since the anonymizing operations are irreversible, once we put a tuple's value into a broader range via sanitization operations, for the remaining participants in downstream, no one can change it back, or even put it in a narrower range. Therefore, the processing effects of previous participants are accumulated.

Simply maximizing a table's utility subject to a current privacy requirement is not a good option: under a lower privacy requirement, merging a pair of tuples may achieve a higher utility, but later, in a higher privacy condition, it turns out that they should be placed in different equivalent classes. This problem not only happens when using our privacy metric, but also exists when using the classic l-diversity or t-closeness standards. For example, in Fig. 2, tuples 3 and 4 are closer to each other, and it is optimal to put them together for 2-diversity conditions. Once it is done, their attributes become the same except for the sensitive attribute column. However, distances from the modified tuples to their second closest tuples are increase d (e.g. the distance between tuples 2 and 3 in T_1), which may cause extra utility losses for creating tables with a higher privacy requirement. Clearly, when sequentially creating a series of tables with increasing privacy protection, tuples grouping orders affect the overall utilities.

KEY OBSERVATIONS

Observation 1: After sanitization, each type of sensitive values must appear at least once in each equivalent class.

According to the definition of our privacy disclosure function P (·), if an equivalent class does not contain a sensitive value, its Jensen-Shannon divergence distance becomes undefined, which breaks the privacy requirement of our problem: $P(T_i) < (l - i + 1)\delta$. Note that Jensen-Shannon divergence is based the Kullback-Leibler divergence. If one type of value appears zero times in one distribution, then we will face the "Division by zero" problem.

Observation 2: In general, the larger a subset of records is, the more likely that its sensitive values' distribution is close to the global one.

Based on Observation. 2, we infer that, if the privacy-preserving level of a set of records is far beyond the current requirement, one can partition the set into multiple subsets to reduce each subset's privacy-preserving level; in the meanwhile, the utility loss of using sanitization operations on the given records can also be reduced.

Observation 3: For a given set of records, whose sensitive values' occurrence times follow a certain distribution, if we partition the given set into several subsets such that the sensitive values' occurrence frequencies in each subset are unchanged, then the amount of privacy disclosure of each subset is the same as the original set's privacy disclosure.

When we use suppression/generalization on a set of records, the larger the set is, the more data utility that is lost. After finding a set of records satisfying a given privacy requirement, one can further reduce the cost of sanitization operations by dividing the set into smaller ones while keeping the distribution of sensitive values unchanged.

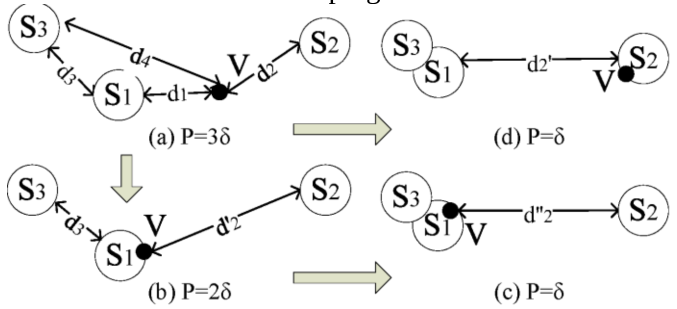


Figure 3 The grouping problem with BUA. On the graph, we assume that S1, S2 and S3 are three large sets of data points, and V is a small set. We use the location distance d between a pair of nodes to represent the difficulties for merging the corresponding data into an equivalent class. In Fig.(a), d1 < d2 < d4 < d1+d3.

Observation 4: For a given set of records, one can create several subsets by letting each subset uniquely dominate on one or more sensitive value types. The merging of these subsets may reduce the statistical distance between their sensitive values' distribution and the global one.

For instance, given a set of records, whose sensitive values' occurrence times are 3 : 5 : 7 : 9 : 11, we can create two subsets with sensitive values' occurrence times 1 : 1 : 6 : 1 : 1 and 1 : 1 : 1 : 8 : 1, respectively. The JS distances of these subsets to the original set are 0.6137 and 0.5994. After merging sets into one equivalent class, the occurrence times of sensitive values become 2 : 2 : 7 : 9 : 2, whose JS distance becomes 0.3075.

SOLUTION DETAILS

This section consists of three parts. In part one, we propose a bottom-up table sanitization approach. The main idea of the bottom-up approach is to merge similar tuples into an equivalent class until the table meets the privacy requirement. By tracking any tuple's locations in the table series, we can see that, with the growth of privacy-preserving level, the size of the tuple's resided equivalent class becomes larger and larger. In the second part, we propose a top-down table sanitization approach. Unlike in the previous approach, the top-down approach gives more concern on the table creation at higher privacy-levels. The intuition of the second approach is to gradually partition tables into multiple clusters until any further partitioning will break the privacy requirement. In order to save more data utility, after reaching the required privacy-level, both the bottom-up approach and top-down approach split the equivalent classes into smaller ones by using observation 3's feature. In the last part, we first compare the above two schemes, and then propose another approach by considering the utility loss at future higher privacy requirements. Note that the first two algorithms only consider the instant privacy requirement (single level), while the last one takes both current and future privacy requirements (multiple levels) into consideration.

Algorithm 1 Bottom-Up table sanitization Approach (BUA)

```
1: Input:Privacy requirement threshold $\delta$ and original table $T$
2: /*Phase I: Create sensitive-value dominating groups*/
3: Create basic group set $\{g\}$ by including each type of sensitive value once
4: Compute semantic distance $d$ from any un-grouped tuple to $\{g\}$
5: for Each basic group $g_i \in \{g\}$ do
6:     for Each type of sensitive value $a_i \in A_n$ do
7:         Compute the overall distance from $g_i$ to the tuples with $a_i$
8: Create data fragments $\{f\}$ dominating one sensitive value type.
9: /*Phase II: Group combinations*/
10: while $\exists f_i \in \{f\}$, whose $P(f_i) > \delta$ do
11:     for Each fragment $f_j \in \{f\}$, and $f_j \neq f_i$ do
12:         Compute the distance $d(f_i, f_j)$ for merging $f_i$ and $f_j$
13:     Sort $d(f_i, f_j)$ in ascending orders
14:     for Each value in $d(f_i, f_j)$ do
15:         if $P(\{f_i, f_j\}) < P(f_i)$ then
16:             merge $f_i$ and $f_j$ into one equivalent class
17: Create grouping index $IDX$ based on $\{f\}$
18: /*Phase III: Further partitioning based on Observation 3*/
19: Call Algorithm. 2, $IDX \leftarrow PLSA(\delta, T, IDX)$
    Output:The grouping index set $IDX$
```

Algorithm 2 Privacy Level-preserved Splitting Alg. (PLSA)

```
1: Input:Privacy requirement threshold $\delta$, original table $T$, and grouping index $IDX$
2: Setup: operating table set $S \leftarrow IDX$ and $IDX \leftarrow \phi$
3: for $S \neq \phi$ do
4:     Fetch table $T'$ from $S$, $S \leftarrow S \setminus \{T'\}$
5:     for Each type of sensitive value $a_i \in A_n$ in $T'$ do
6:         Assign tuples with $a_i$ into two groups $S_{a_i}$ and $S'_{a_i}$
7:     Create subtables $T_1$, $T_2$ by optimally combining $\{S_{a_i}\}$, $\{S'_{a_i}\}$
8:     if $P(T_1) < \delta$ and $P(T_2) < \delta$ then
9:         Update the operation set $S \leftarrow S \bigcup \{T_1, T_2\}$
10:     else
11:         Update the index set $IDX \leftarrow IDX \bigcup \{T'\}$
    Output:The grouping index set $IDX$
```

A. *Bottom-up Approach (BUA)*

The Bottom-up Approach (BUA) treats each record individually, and the table creating process by BUA is similar to the changing pattern of tuples' grouping during multi-hop relays: initially, there are lots of small-sized equivalent classes, and gradually, more and more classes merge into others; with the growth of the sizes of equivalent classes, the distribution of the sensitive values within each equivalent class becomes closer to the distribution over the whole table.

There are two objections with our problem: on the one hand, we need to maximize a table's remaining utility after sanitization operations, and on the other hand, the amount of privacy disclosure of the resulting table should be bounded. Since privacy and utility are not comparable with each other, we cannot consider these two objections at the same time. In order to meet the utility objection, one can merge similar tuples into one equivalent class, as many as possible, until the class reaches the required privacy-level. For satisfying the privacy, one can group the tuples according to the occurrence frequency of the sensitive values. For example, if the whole table's occurrence frequencies of the sensitive values are 20% : 40% : 40%, then when creating an equivalent class, the operator should merge at least 5 tuples in each time, including one tuple with sensitive value type I, two tuples with type II, and another two tuples with type III. However, neither of the approaches can create a table with maximum utility and satisfy a given privacy-preserving level.

The basic idea of BUA is as follows: at the lower privacy-preserving requirements, if each equivalent class dominates in one type of sensitive value, the merging of different classes may make their sensitive values' distribution closer to that of the whole table. BUA consists of two phases. Considering that each type of sensitive values must occur once in each equivalent class, in phase I, BUA first creates several basic groups, where each sensitive value appears once within each group. For the remaining records, we compute their quasi-identifiers' distance to each group. BUA proportionally assigns each basic group with a dominating sensitive value type, according to the occurrence frequency of sensitive values. Based on tuple's sensitive value and its distance to groups, each tuple joins a group. In phase II, BUA gradually merges groups in order to satisfy a given privacy requirement, and the merging process will not terminate until all equivalent classes reach the required privacy-level. Algorithm. 1 gives the procedure of BUA.

Essentially, by using BUA, along the propagation path, each node tries to locally maximize its table's utility subject to the privacy requirement. However, the process may cause inappropriate grouping, especially when creating the basic groups. As shown by Fig. 3, initially data point V is slightly closer to equivalent class S_1 than class S_2. At a lower privacy requirement (Fig. 3(b)), BUA should merge V with S_1, and in a later time (Fig. 3(c)), the S_1 will merge with another equivalent class S_3. However, if we directly increase the privacy requirement from 3δ to δ, the data point V will join the S_2 set. In short, whether a node joins an equivalent class is determined not only by the attribute distance between the node and the class, but also the distance between the node and other classes at which the class is going to merge at higher privacy requirements. However, BUA does not consider the impacts of further merging operations.

Algorithm 3 Top-down table sanitization Approach (TDA)

```
1: Input:Privacy requirement threshold $\delta$ and original table $T$
2: /*Phase I: Clustering until any further clustering violates $\delta$*/
3: Set up the operation table set by $S \leftarrow \{T\}$
4: Set up output equivalent class index set by $IDX \leftarrow \phi$
5: while $S \neq \phi$ do
6:     Fetch a table $T'$ from $S$, $S \leftarrow S \setminus \{T'\}$
7:     Create 2 clusters $T_1$ and $T_2$: $T_1 \bigcap T_2 = \phi$ and $T_1 \bigcup T_2 = T'$
8:     if $P(T_1) < \delta$ and $P(T_2) < \delta$ then
9:         Update the operation set $S \leftarrow S \bigcup \{T_1, T_2\}$
10:    else
11:        Update the index set $IDX \leftarrow IDX \bigcup \{T'\}$
12: /*Phase II: Further partitioning based on Observation 3*/
13: Call Algorithm. 2, $IDX \leftarrow PLSA(\delta, T, IDX)$
    Output:The grouping index set $IDX$
```

B. *Top-down Approach (TDA)*

Top-down Approach (TDA) considers how to partition a table into multiple sub-tables such that each sub-table satisfies a single-level privacy requirement. The basic idea of TDA comes from paper [23]. Our TDA is based on two insights. First, for a given set of data {S}, the distribution of which follows D({S}), if we partition the set {S} into two subsets ({S1}, {S2}) and keep each value's occurrence frequency unchanged in both of the subsets, then we have

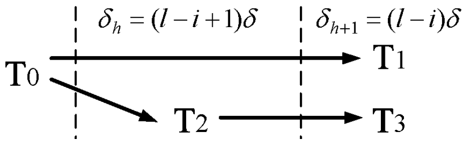
D({S1}) ≅ D({S}) and D({S2}) ≅ D({S}). This feature maximizes the protection of privacy but hurts data utility. The second insight is that, if we randomly partition a table into several sub-tables, the larger a sub-table is, the more likely that the distribution of a sensitive attribute in the sub-table is close to the distribution of the attribute in the original table.

Based on these two insights, we propose our TDA, as shown in Algorithm 3. The algorithm consists of two phases. The first phase gradually partitions the original data table into several sub-tables. This part tries to maximally preserve data utility and satisfy the needs of privacy. Note that, during the process of the table's propagation, once a pair of records been merged into one equivalent class at a node, it is hard for the following nodes to take them apart, since the following nodes cannot discriminate the tuples from the same equivalent class. Based on this consideration, instead of partitioning the original table at one time, we first divide the table into two parts, then we split each sub -table into another two parts, and so on. For any sub-table, the partitioning process stops when the further partitioning on it will break the privacy requirement. Due to the fact that the smaller an equivalent class is, the more data utility preserves, the second phase tries to further reduce the size of each sub-table T, and in the meanwhile, keeps their privacy-preserving degree unchanged by using Algorithm 2.

C. Forward Looking Approach (FLA)

In order to achieve the optimization of the overall utility of tables, we propose another algorithm called Forward Looking-based table sanitization Approach (FLA), which improves the solutions from two aspects. For reducing the impacts of locality, instead of only considering the current privacy requirement, FLA takes the next privacy-level into account: FLA creates two tables T_i and T_{i+1} such that $U(T_i) + U(T_{i+1})$ is maximized subject to $P(T_i) < (l - i + 1)\delta$ and $P(T_{i+1}) < (l - i)\delta$.

FLA also improves the results from methodology. The advantage of BUA is that, by grouping tuples into fragments dominating on certain sensitive values, one can easily create tables with different privacy-levels. However, BUA's main problem is caused by the inappropriate merging at the lower privacy requirements, which results in lots of utility loss at the conditions with higher privacy requirements. On the other hand, TDA can easily group tuples at higher privacy requirements, but its phase II lacks flexibility: any partition on a sub-table will not change the sensitive values' distribution, which potentially causes utility loss.


$\delta_h = (l - i + 1)\delta$
$\delta_{h+1} = (l - i)\delta$
T0
T1
T2
T3
Figure 4 Forward looking the future privacy requirement.

FLA combines both TDA and BUA together. Suppose that we have a table T_0 and two privacy thresholds (l−i+1)δ and (l−i)δ, where 1 < i ≤ l. From T_0, FLA directly creates two tables T_1 and T_2 with maximal utilities, where $P(T_1) < (l - i)\delta < (l - i + 1)\delta$ and $P(T_2) < (l - i + 1)\delta$. Based on T_2, FLA further builds up another table T_3such that $P(T_3) < (l - i)\delta$, as shown by Fig. 4. Next, FLA compares the utilities between 2 × $U(T_1)$ and $U(T_2)$ + $U(T_3)$. If the former is greater, then FLA will publish T1 under the privacy requirement as $(l - i + 1)\delta$; otherwise, it will publish T_2 .

When creating a table T_i with privacy level δ_h, FLA adopts TDA Phase I to create server master equivalent classes, and within each master class, FLA compares the utility loss by using BUA or TDA Phase II. Since BUA is more flexible for satisfying the privacy requirement, FLA adopts TDA Phase I to partition data into multiple sub-tables, and within each sub-table, FLA uses BUA and TDA to create equivalent classes. The final equivalent classes come from the partitions with maximal utility.

During TDA Phase I, we strictly use binary partitioning, as shown by Fig. 5. The reason is that at a lower privacy-level, one may create several clusters, for instance, assuming 3 clusters, and then, at a higher privacy-level, the number of clusters becomes less, assuming 2 clusters. Clearly, one cluster in the lower privacy-level has to split and merge with others in order to achieve a higher privacy-level later. Many data utilities will be lost while splitting an equivalent class, especially when the size of the equivalent class is large. Usually, the higher the privacy-level is, the bigger an equivalent class is. Although during a table's propagation, tuples are clustering from lower privacy-levels to higher levels, during the top-down splitting process, the partitioning results at the earlier phases satisfy the higher-level requirements, and then with the reducing of their sizes, they can no long meet the higher-level requirements.

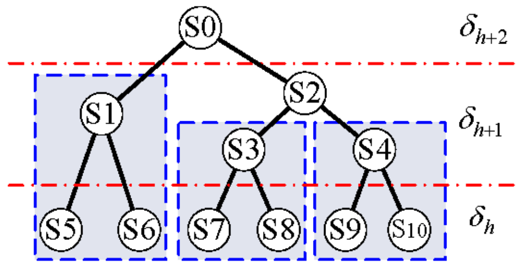
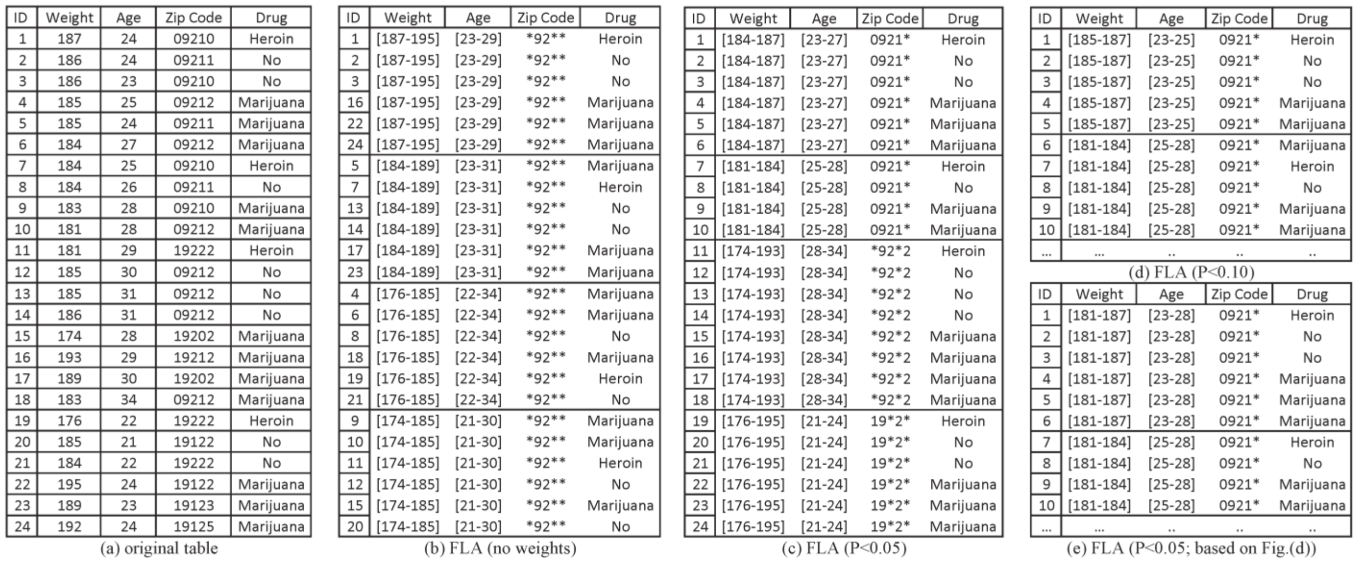


Figure 5 **The idea of FLA. In the figure $\delta_{h+2} < \delta_{h+1} < \delta_h$**

For achieving a single privacy requirement δ_h, FLA first gradually partitions the table into several blocks such that each block meets the current privacy level. For example, in Fig. 5, data sets S5 to S10 represent these blocks. Next, within their parent blocks, S1, S3, and S4, FLA uses BUA to create equivalent classes. Finally, FLA compares the results of both BUA and TDA Phases, and adopts the best one as the equivalent class.

D. Example

Fig. 6 gives an example. Fig. 6 (a) is the original table, which consists of 24 tuples. The attributes: “Weight”, “Age”, and “Zip Code” are the quasi-identifiers, the combination of which ma y unique identify a person. “Drug” is the sensitive attribute, which has three types of value: “Heroin”, “Marijuana”, and “No”. The occurrence frequencies of these three types of value follows “Heroin”:“No” :“Marijuana” = 1 : 2 : 3.



ID	Weight	Age	Zip Code	Drug
1	187	24	09210	Heroin
2	186	24	09211	No
3	186	23	09210	No
4	185	25	09212	Marijuana
5	185	24	09211	Marijuana
6	184	27	09212	Marijuana
7	184	25	09210	Heroin
8	184	26	09211	No
9	183	28	09210	Marijuana
10	181	28	09212	Marijuana
11	181	29	19222	Heroin
12	185	30	09212	No
13	185	31	09212	No
14	186	31	09212	No
15	174	28	19202	Marijuana
16	193	29	19212	Marijuana
17	189	30	19202	Marijuana
18	183	34	09212	Marijuana
19	176	22	19222	Heroin
20	185	21	19122	No
21	184	22	19222	No
22	195	24	19122	Marijuana
23	189	23	19123	Marijuana
24	192	24	19125	Marijuana

(a) original table

ID	Weight	Age	Zip Code	Drug
1	[187-195]	[23-29]	*92**	Heroin
2	[187-195]	[23-29]	*92**	No
3	[187-195]	[23-29]	*92**	No
16	[187-195]	[23-29]	*92**	Marijuana
22	[187-195]	[23-29]	*92**	Marijuana
24	[187-195]	[23-29]	*92**	Marijuana
5	[184-189]	[23-31]	*92**	Marijuana
7	[184-189]	[23-31]	*92**	Heroin
13	[184-189]	[23-31]	*92**	No
14	[184-189]	[23-31]	*92**	No
17	[184-189]	[23-31]	*92**	Marijuana
23	[184-189]	[23-31]	*92**	Marijuana
4	[176-185]	[22-34]	*92**	Marijuana
6	[176-185]	[22-34]	*92**	Marijuana
8	[176-185]	[22-34]	*92**	No
18	[176-185]	[22-34]	*92**	Marijuana
19	[176-185]	[22-34]	*92**	Heroin
21	[176-185]	[22-34]	*92**	No
9	[174-185]	[21-30]	*92**	Marijuana
10	[174-185]	[21-30]	*92**	Marijuana
11	[174-185]	[21-30]	*92**	Heroin
12	[174-185]	[21-30]	*92**	No
15	[174-185]	[21-30]	*92**	Marijuana
20	[174-185]	[21-30]	*92**	No

(b) FLA (no weights)

ID	Weight	Age	Zip Code	Drug
1	[184-187]	[23-27]	0921*	Heroin
2	[184-187]	[23-27]	0921*	No
3	[184-187]	[23-27]	0921*	No
4	[184-187]	[23-27]	0921*	Marijuana
5	[184-187]	[23-27]	0921*	Marijuana
6	[184-187]	[23-27]	0921*	Marijuana
7	[181-184]	[25-28]	0921*	Heroin
8	[181-184]	[25-28]	0921*	No
9	[181-184]	[25-28]	0921*	Marijuana
10	[181-184]	[25-28]	0921*	Marijuana
11	[174-193]	[28-34]	*92*2	Heroin
12	[174-193]	[28-34]	*92*2	No
13	[174-193]	[28-34]	*92*2	No
14	[174-193]	[28-34]	*92*2	No
15	[174-193]	[28-34]	*92*2	Marijuana
16	[174-193]	[28-34]	*92*2	Marijuana
17	[174-193]	[28-34]	*92*2	Marijuana
18	[174-193]	[28-34]	*92*2	Marijuana
19	[176-195]	[21-24]	19*2*	Heroin
20	[176-195]	[21-24]	19*2*	No
21	[176-195]	[21-24]	19*2*	No
22	[176-195]	[21-24]	19*2*	Marijuana
23	[176-195]	[21-24]	19*2*	Marijuana
24	[176-195]	[21-24]	19*2*	Marijuana

(c) FLA (P<0.05)

ID	Weight	Age	Zip Code	Drug
1	[185-187]	[23-25]	0921*	Heroin
2	[185-187]	[23-25]	0921*	No
3	[185-187]	[23-25]	0921*	No
4	[185-187]	[23-25]	0921*	Marijuana
5	[185-187]	[23-25]	0921*	Marijuana
6	[181-184]	[25-28]	0921*	Marijuana
7	[181-184]	[25-28]	0921*	Heroin
8	[181-184]	[25-28]	0921*	No
9	[181-184]	[25-28]	0921*	Marijuana
10	[181-184]	[25-28]	0921*	Marijuana
...	...	..	..	..

(d) FLA (P<0.10)

ID	Weight	Age	Zip Code	Drug
1	[181-187]	[23-28]	0921*	Heroin
2	[181-187]	[23-28]	0921*	No
3	[181-187]	[23-28]	0921*	No
4	[181-187]	[23-28]	0921*	Marijuana
5	[181-187]	[23-28]	0921*	Marijuana
6	[181-187]	[23-28]	0921*	Marijuana
7	[181-184]	[25-28]	0921*	Heroin
8	[181-184]	[25-28]	0921*	No
9	[181-184]	[25-28]	0921*	Marijuana
10	[181-184]	[25-28]	0921*	Marijuana
...	...	..	..	..

(e) FLA (P<0.05; based on Fig.(d))

Figure 6 Example. The value of tuples 11 − 24 in both Figs. (d) and (e) are the same as them in Fig. (c).

Our proposed algorithms not only work on the condition that each attribute has a unique weight (also known as source-based utility weights), but can be also applied under the scenario in which tuples have personalized weights. In Fig. 6 (a), assume that there is a background knowledge in which tuples 1 to 10 are Hollywood stars, and tuples 11 to 24 are vagrants. Consider the fact that Hollywood stars care a lot about their weights and some of them use drugs, such as heroin, to lose weight. The “Weight” attribute is important for these groups of data, but it is not for the vagrants. Instead, the living area is an important factor about drug using habits. Therefore, during the creation of equivalent classes, one may give different weights to different types of users' data. In this example, we give 0.5 utility weight to “Weight” and another 0.5 weight to “Age” for the Hollywood group, while assigning 0.5 utility weight to both “Zip Code” and “Age” attributes for the vagrant group. Here, we let the table be propagated twice, and let δ = 0.05. So, after the first time of transmission, the privacy disclosure of the result table should smaller than 2δ = 0.1, and after the second time, it should be less than 0.05. Figs 6 (b) and (c) show the sanitation results with and without using the personalized weights. Since the personalized weights put similar users' tuples closer regarding the overall attribute distance, Fig 6 (c) has more data utility than (b). Note that when using the personal weights the attribute distance is no longer symmetric.

When using FLA preparing for the first time of propagation, FL A creates two clusters: tuples $1 - 10$ and $11 - 24$. Since their privacy disclosure amount is 0.0069 and 0.0038, which are both smaller than the privacy requirement, FLA continues splitting the clusters into tuple groups $1 - 5$, $6 - 10$, $11 - 18$, and $19 - 24$. The four groups' privacy disclosure amount is 0.0292, 0.667, 0.0122, and 0 (as shown by Fig.(d)). Since any further partitioning breaks the privacy restriction, Fig.(d) is the result by directly using TDA with privacy requirement as 0.1. However, FLA also considers the utility cost of future sanitization, and therefore, FLA computes Figs.(c) and (e). In Fig.(c), FLA directly adopts TDA to create a table for the future privacy level $P < 0.05$, and based on Fig.(d), FLA builds a table for the next privacy-level, as shown by Fig.(d). After computing the utility loss of the results, FLA finds out that by directly using Fig.(c) under the privacy requirements, $P < 0.1$ and $P < 0.05$ can save slightly more data utility than using Figs.(d) and (e). FLA takes Fig.(c) as final result.

SECURITY ANALYSIS

In this paper, we consider an honest but curious attacking model, in which each participant of the system could be an attacker. More specifically, in this model, the attackers will always follow the rule for providing correct information to the next participant on a social chain. However, the attackers may be willing to learn more information than he supposes to do. Note that, in practice, the information propagation paths form a graph, instead of a chain, and therefore, the information that an attack is able to obtain only comes from his direct neighbors. In our paper, a system is privacy-preserving if, for any participant, the overall information that he could learn from all of his neighbors is no greater than the information that he directly obtained from the neighbor, who has the closest distance to the source node v0. For the ease of description, let $\|u - v\|$ be the social distance between users u and v, which is counted as the number of hops along the shortest path between them, $\|u - v\| = \|v - u\|$. Let T_{v_i} be the table possessed by user v_i and $H(A_n|T_{v_i})$ be the conditional entropy that one learns about the sensitive attribute based on table T_{v_i}.

Theorem 4: Assume v_i, v_j are two nodes along the same information propagation path from v_0. If all participants adopt the same δ and attribute weights$\{W\}$, and$\|v_i - v_0\| < \|v_j - v_0\|$, then v_j's table must be a sanitization result of v_i's table.

Proof: In the proposed scheme, along the information propagation path, the participants gradually hide more and more information by putting tuple values into a broader ranger. The participant at downstream takes the output from his previous participant as its input during sanitization operations. Since sanitization is irreversible, there exists a series of participants, $v_{i+1}, v_{i+2}, \ldots, v_{i+(j-i-1)}$ such that any table belonging to them is the sanitization result of its previous table. Therefore, v_j's table must also be a sanitization result of v_i's table. ■

Theorem 5: If all participants adopt the same δ and attribute weights $\{W\}$, for any pair of participants v_i and v_j with $\|v_i - v_0\| = \|v_j - v_0\|$,, we have $T_i = T_j$ by using FLA.

Proof: Each step in FLA is determined: for the same input table T and privacy requirement δ, the output of FLA is always the same. Note that, when using FLA, even if two tuples are identical, FLA uses the row numbers to provide a unique operation order to them. As a result, there is no stochastic event in FLA. Now, consider two shortest information propagation paths, assuming $\{v_1, v_2, \ldots, v_l\}$ and $\{v'_1, v'_2, \ldots, v'_l\}$. Since all participants use the same δ and the original data owner v_0 gives the same table to the first receiver on each chain, the outputs of v_1 and v'_1 are the same. By induction, we know that whenever $\|v_i - v_0\| = \|v'_i - v_0\|$, the sanitization results on v_i and v'_i will be the same. Therefore, for the same initial owner v_0 and parameters, the solution of FLA only relates with the shortest hop numbers from its executer to v_0. ■

Definition 3: A scheme is called anti-colluding if the total information gain from the tables of the colluding users, $H(A_n|T_{v_1}, T_{v_2}, \ldots)$, is not greater than cahoot's maximum information gain $max_i H(A_n|T_{v_i})$.

Note that colluding attackers may not be direct neighbors with each other. Let T_{v_i} be the table that an attacker obtained from his social chain. Tables T_{v_i} and T_{v_j} may be created on different social chains.

Theorem 6: For a given initial table T_0 at v_0, FLA is naturally against colluding attacks, when all participants adopt the same δ and attribute weights $\{W\}$,.

Proof: For a given threshold δ and propagation length l, the amount of disclosed information is only related to the recipient's distance toward v_0. According to theorem 4, if the distance between v and source is smaller than that of v' , then the obtained information of v' ' is a subset of that of v. For a group of cahoots, the maximum information that they could obtain must come from the member with the least distance toward the source v_0. As a result, our scheme is against colluding attacks. ■

EVALUATION

In this section, we conduct extensive real data-based simulations to evaluate the performances of our scheme. For the ease of comparison, we use black bars to represent the results by using the Bottom-Up Approach (BUA in Section V.A); We use grey shadowed bars to represent the results of the Top-Down Approach (TDA in Section V.B). The white shadowed bars stand for the results of the Forward Looking Approach (FLA in Section V.C), which compares the following utility wastes of the best and the second best merging options.

A. Simulation Setup and Evaluation Metric

In the simulation, we use the Statlog (German Credit Data) Data Set, which consists of 1, 000 instances and 25 attributes. For the ease of simulation, we use the file named "german.data -numeric", which coded categorical attributes as integers. We let the first column be the sensitive attribute, and let the remaining 24 attributes be the quasi-identifiers. During simulation, we consider the weighted Euclidean distance between tuples as their merging costs:

$$d(\sigma_i(T), \sigma_j(T)) = \sqrt{\sum_{k=1}^{n-1} w_i \times \left(\sigma_i\left(\pi_k(T)\right) - \sigma_j(\pi_k(T))\right)^2}$$

where w_i is the attribute weight, $\sum w_i = 1$. In practice, this distance could be computed as hamming code distance or the number of modified digits by using suppression. After creating an equivalent class, the values of its members' quasi-identifiers will be replaced with their mean value within the class.

Instead of directly computing the remaining utility of each table, we count the percentage of data utility that has been wasted after sanitization. Let $\sigma_i(T')$ be the members from the same equivalent class after a sanitization operation, and $\sigma_i(T)$ be their corresponding original data, then the utility loss UL is defined as following:

$$UL\left(\sigma_i(T), \sigma_i(T')\right) = 1 - \frac{d(\sigma_i(T'), \sigma_i(\tilde{T}))}{d(\sigma_i(T), \sigma_i(\tilde{T}))}$$

where $\tilde{\tilde{T}}$ stands for the resulting table by sanitizing the initial table T_0 into one equivalent class, $\tilde{\tilde{T}} = \pi(T_0)$.

B. Simulation Results

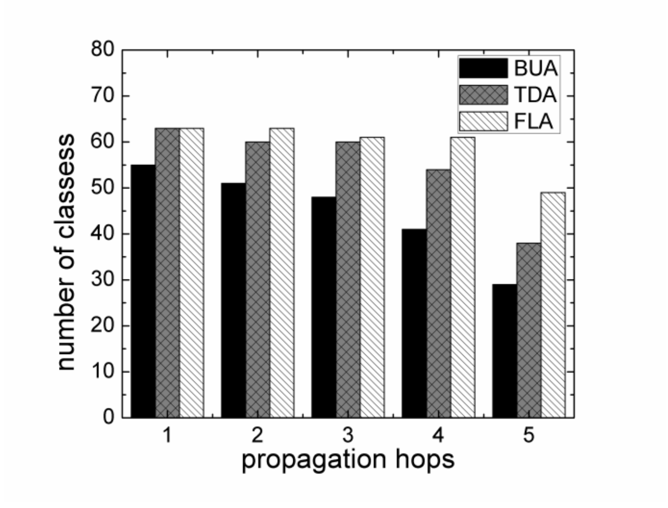


Figure 7 The number of created equivalent classes by different approaches.

Fig.7 shows the changing pattern of the number of created equivalent classes at each participating nodes during the table's propagation. Note that, the closer a participating node is to the source, the greater the corresponding table's utility that is preserved. As Fig.7 shows, the number of equivalent classes decreases along the information propagation path. However, the reducing speed is not linear. As we have mentioned, the maximum number of equivalent class is bounded by the number of sensitive values that have the minimum occurrences. Therefore, in the first hop, TD A and FLA create the same number of equivalent classes. During the propagation, more and more tuples merge into a larger group, and form a new equivalent class. Since certain approaches may inappropriately put some tuples into an equivalent class at an earlier time, the creation of equivalent classes with higher privacy-preserving requirements becomes harder and harder. As a result, they may build less equivalent classes than others at the later time.

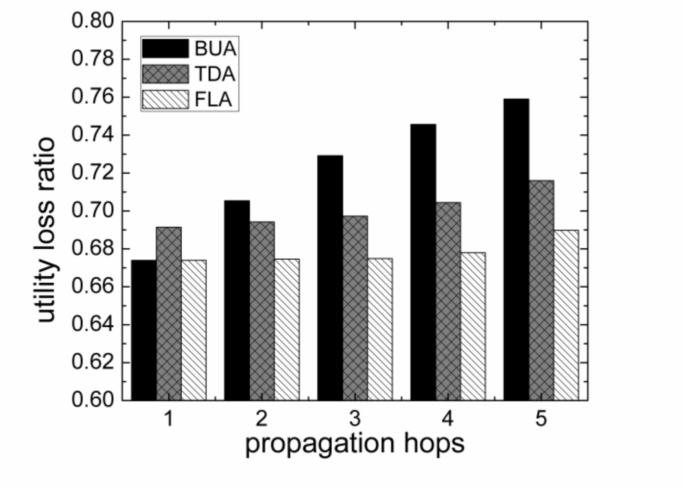


Figure 8 Average utility loss during propagation

The system's overall utility loss is shown by Fig. 8. In this figure, the x-axis indicates the total number of propagated hops, and the y-axis gives the average utility loss per table. From the figure, we can see that the average amount of utility loss increases with the growing length of propagations. FLA has the smallest growing speed, while BUA has the largest one. In general, FLA preserves more data utilities than the other two approaches.

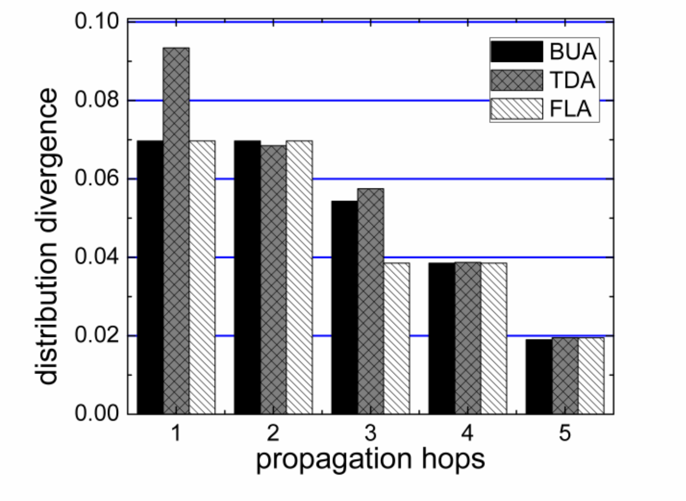


Figure 9 Average privacy of each table

During the table propagation, although all result tables satisfy the same set of privacy requirements, the exact privacy-preserving values are different. Fig. 9 shows the changing pattern of each tables' privacy values. The horizontal lines show the increasing privacy requirements. The y-axis represents the privacy-level of each created table along a propagation path, and we measure the privacy-level by function (T) from section II. In most cases, BUA preserves more privacy than TDA. Since FLA considers both the current and future privacy requirements, at hop 3 in Fig. 9, it directly uses the privacy requirement of the fourth hop.

Fig. 10 gives the pattern of utility loss with an increase in privacy-disclosure threshold δ. Note that, in the previous three simulation, gradually create a serial of tables with increasing privacy-preserving levels: the input at hop i is the output of hop $i-1$. However, in Fig. 10, we always use the original table T_0 as input, and assign different privacy-preserving level δ. From the figure, we can see that the amount of utility loss dramatic ally increases at the higher privacy-preserving threshold (a smaller δ). As we expected, the tables created by BUA loses the most data utility, since its tuple-merging scheme (at an earlier phase) may cause inappropriate overall clustering results. We can also find that, at the condition of lower privacy-preserving thresholds, there are no significant differences between the TDA and FLA. The main reason for this phenomenon is that the members of each class created by these two approaches are basically similar to each other at the lower thresholds.

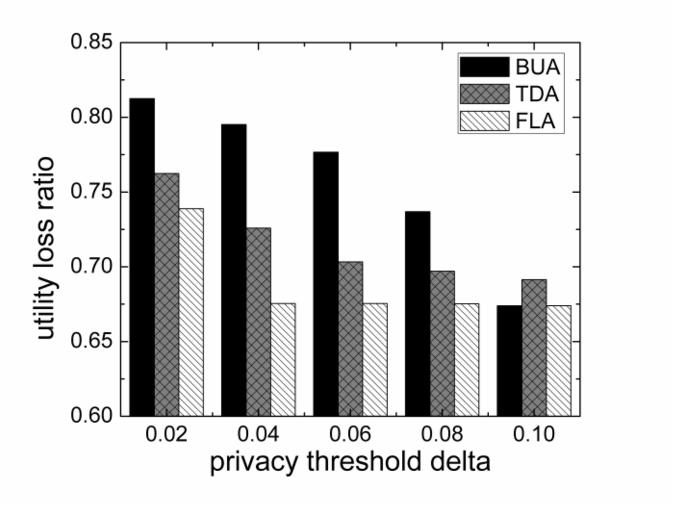


Figure 10 Utility lost by different approaches

Finally, we consider the computing complexities of these three methods in Fig. 11. Note that the solution space of the attribute-level approach is in $O(n^2)$ while that of the tuple-level approaches is in $O(m^2)$, where m is the number of records, n is the attributes, and $m \gg n$. As a result, the computing speed of the attribute-level approach is much faster than the approaches at a tuple-level. Since the looking forward method uses both bottom-up and top-down approaches multiple times and creates several tables at different privacy levels, it takes a significant time to find tuples' merging results.

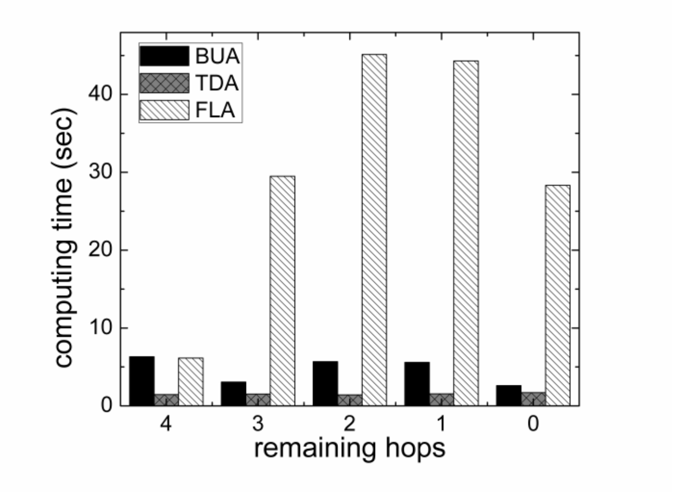


Figure 11 Computing time at each node

CONCLUSION

Privacy-preserving data publishing is a popular research problem. Based on the unique features of data tables, many specific approaches have been designed. K-anonymity, l-diversity, and t-closeness are the three well-known sanitization standards for publishing microdata tables. However, to our best knowledge, all of the existing studies focus on providing a single level privacy protection via a centralized scheme. Consider that the table's recipients are not equally trustable, especially on a social network. In this paper, we propose a new research problem: given a source node and a social path with length l, how are we to sequentially generate l tables, such that their overall utility is maximized and data privacy is gradually enhanced along the propagation path. For solving the problem, one has to consider the similarity between individual records, together with the sensitive values' distribution within each equivalent class. Clearly, the problem is more complex than the conventional optimization problems of K-anonymity or t-closeness. In this paper, we first design two local algorithms for optimally sanitizing tables with a single privacy-requirement, and then, we combine these two algorithms together and use a privacy requirement forward looking scheme: during sanitization, instead of minimizing the utility loss under the current privacy-level, our scheme considers the sanitization impacts on the future. Extensive simulation results show that our proposed algorithms can successfully increase the overall data utility, and meet the enhancing privacy-preserving requirements.

REFERENCES

[1] Wu, J., & Wang, Y. (2012). Social feature-based multi-path routing in delay tolerant networks. In *Proceedings of the 31st Annual IEEE International Conference on Computer Communications* (pp. 1368-1376). Orlando, Florida, USA.
[2] Feng, W., & Wang, J. (2012). Incorporating heterogeneous information for personalized tag recommendation in social tagging systems. In *Proceedings of the 18th* ACM SIGKDD *international conference on Knowledge discovery and data mining* (pp. 1276-1284). Beijing, China.
[3] Aiello, L. M., Barrat, A., Schifanella, R., Cattuto, C., Markines, B., & Menczer, F. (2012). Friendship prediction and homophily in social media. *ACM Transactions on the Web*, 6(2), 9.
[4] Meyerson, A., & Williams, R. (2004). On the complexity of optima l k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems* (pp. 223-228). Paris, France
[5] Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based* Systems, 10(5), 557-570.
[6] Machanavajjhala, A., Kifer, D., Gehrke, J., & Venkitasubramaniam, M. (2007). l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data* 1(1), 3.
[7] Li, N., Li, T., & Venkatasubramanian S. (2007). t-Closeness: Privacy Beyond k-Anonymity and l-Diversity. In *Proceedings of IEEE 23rd International Conference on Data Engineering* (pp.106-115). Istanbul, Turkey.
[8] Chawla, S., Dwork, C., McSherry, F., Smith, A. & Wee, H. (2005). Toward privacy in public databases. In J. Kilian (Ed.) *Theory of Cryptography* (pp. 363-385). Springer.
[9] Dwork, C. (2011). A firm foundation for private data analysis. *Communications of the ACM 54*(1), 86-95.
[10] Roth, A., & Roughgarden, T. (2010). Interactive privacy via the median mechanism. In *Proceedings of the 42 ACM Symposium on Theory of Computing* (pp. 765-774).
[11] Blum, A., Ligett, K., & Roth, A. (2013). A learning theory approach to non-interactive database privacy. *Journal of the ACM 60*(2), 12.
[12] Dwork, C. (2006). Differential privacy. *Theory of cryptography* (pp. 496-502). Springer.
[13] Nissim, K., Smorodinsky, R., & Tennenholtz, M. (2012). Approximately optimal mechanism design via differential privacy. In *Proceedings of the 3rd ACM Innovations in Theoretical Computer Science Conference* (pp. 203-213).
[14] Samarati, P. (2001). Protecting respondents identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6),1010-1027.
[15] Kisilevich, S., Rokach, L., Elovici, Y., & Shapira, B. (2010). Efficient multidimensional suppression for k-anonymity. *IEEE Transactions on Knowledge and Data Engineering,* 22(3), 334-374
[16] Sweeney, L. (2002). Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* 10(5), 571-588.
[17] Xiao, X., Tao, Y., & Chen, M. (2009). Optimal random perturbation on at multiple privacy levels. In *Proceedings of the VLDB Endowment* (pp. 814-825). Lyon, France.
[18] Li, Y., Chen, M., Li, Q., & Zhang, W. (2012). Enabling multilevel trust in privacy preserving data mining. *IEEE Transactions on Knowledge and Data Engineering,* 24(9), 1598-1612
[19] Brickell, J., & Shmatikov, V. (2008). The cost of privacy: destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp.70-78), Las Vegas, NV, USA
[20] Li, T., & Li, N. (2009). On the tradeoff between privacy and utility in data publishing. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp.517-526). Paris, France

[21] Lin, J. (1991). Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1), 145-151.
[22] Xiao, X., Yi, K., & Tao, Y. (2010). The hardness and approximate on algorithms for l-diversity. In *Proceedings of the 13th ACM International Conference on Extending Database Technology* (pp. 135-146). Lausanne, Switzerland
[23] LeFevre, K., DeWitt, D., & Ramakrishnan, R. (2006). Mondrian multidimensional k-anonymity. In *Proceedings of the 22nd IEEE International Conference on Data Engineering* . Atlanta, GA, USA