An efficient hash-based algorithm for minimal $k$-anonymity

Xiaoxun Sun Min Li Hua Wang Ashley Plank
Department of Mathematics & Computing
University of Southern Queensland
Toowoomba, Queensland 4350, Australia
Email: {sunx, limin, wang, plank}@usq.edu.au

Abstract

A number of organizations publish microdata for purposes such as public health and demographic research. Although attributes of microdata that clearly identify individuals, such as name and medical care card number, are generally removed, these databases can sometimes be joined with other public databases on attributes such as Zip code, Gender and Age to reidentify individuals who were supposed to remain anonymous. “Linking” attacks are made easier by the availability of other complementary databases over the Internet.

$k$-anonymity is a technique that prevents “linking” attacks by generalizing and/or suppressing portions of the released microdata so that no individual can be uniquely distinguished from a group of size $k$. In this paper, we investigate a practical model of $k$-anonymity, called full-domain generalization. We examine the issue of computing minimal $k$-anonymous table based on the definition of minimality described by Samarati. We introduce the hash-based technique previously used in mining associate rules and present an efficient hash-based algorithm to find the minimal $k$-anonymous table, which improves the previous binary search algorithm first proposed by Samarati.

Keywords: microdata release, hash-based algorithm, $k$-anonymity.

1 Introduction

Several microdata\cite{1} disclosure protection techniques have been developed in the context of statistical database, such as scrambling and swapping values and adding noise to the data while maintaining an overall statistical integrity of the result (Adam & Wortman 1989, Willenborg & DeWaal 1996). However, many applications require release and explicit management of microdata while maintaining truthful information within each tuple. This ‘data quality’ requirement makes inappropriate those techniques that disturb data and therefore, although preserving statistical properties, compromise the correctness of the single pieces of information. Among the techniques proposed for providing anonymity in the release of microdata (Federal Committee on Statistical Methodology 1994) we focus on two techniques in particular: generalization and suppression (in the Statistics literature, this approach is often called recoding), which unlike other existing techniques, such as scrambling or swapping, preserve the truthfulness of the information.

$k$-anonymity is a technique that prevents joining attacks by generalizing and/or suppressing portions of the released microdata so that no individual can be uniquely distinguished from a group of size $k$. There are a number of models for producing an anonymous table. One class of models, called global-recoding (Willenborg & DeWaal 2001), map the values in the domains of quasi-identifier attributes (defined in Section 2) to other values. This paper is primarily concerned with a specific global-recoding model, called full-domain generalization. Full-domain generalization was proposed by Samarati and Sweeney (Samarati & Sweeney 1998, Samarati 2001) and maps the entire domain of each quasi-identifier attribute in a table to a more general domain in its domain generalization hierarchy. This scheme guarantees that all values of a particular attribute in the anonymous table belong to the same domain.

For any anonymity mechanism, it is desirable to define some notion of minimality. Intuitively, a $k$-anonymous table should not generalize, suppress, or distort the data more than is necessary to achieve such $k$-anonymity. Indeed, there are a number of ways to define minimality. One notion of minimality is defined as to generalize or suppress the minimum number of attribute values in order to satisfy a given $k$-anonymity requirement. Such a problem is $NP$-hard (Aggarwal et al. 2005, Meyerson & Williams 2004). As to our model, the notion of minimal full-domain generalization was defined in (Samarati & Sweeney 1998, Samarati 2001) using the distance vector of the domain generalization. Informally, this definition says that a full-domain generalized private table ($PT$) is minimal if $PT$ is $k$-anonymous, and the height of the resulting generalization is less than or equal to that of any other $k$-anonymous full-domain generalization.

In this paper, we focus on this specific global-recoding model of $k$-anonymity. Our objective is to find the minimal $k$-anonymous generalization (table) under the definition of minimality defined by Samarati (Samarati 2001). By introducing the hash-based technique, we provide a new approach to generate minimal $k$-anonymous tables that not only improves the search algorithm proposed by Samarati (Samarati 2001) but is also useful for computing other optimal criteria for $k$-anonymity.

The remainder of this paper is organized as follows. In Section 2, we introduce some notions of $k$-anonymous table. In Section 3, we introduce our hash technique used in this paper. In Section 4, we introduce the generalization relationship and the definition of minimal $k$-anonymous table. Our core hash-based

---

Copyright ©2008, Australian Computer Society, Inc. This paper appeared at the Thirty-First Australasian Computer Science Conference (ACSC2008), Wollongong, Australia. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 74. Gillian Dobbie and Bernard Mans, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

\footnote{The term “microdata” refers to data published in raw, non-aggregated form (Willenborg & DeWaal 2001).}
algorithm and comparisons with previous algorithm discussed in Section 5. Related work is discussed in Section 6. Conclusion and future work are drawn in Section 7.

2 k-anonymous private table

The concept of k-anonymity (Samarati & Sweeney 1998) tries to capture one of the main requirements that has been followed by the statistical community and by agencies releasing data on the private table (PT). According to the requirement, the released data should be indistinguishably related to no less than a certain number of respondents. The set of attributes included in the private table, also externally available and therefore exploitable for linking, is called quasi-identifier (QI). The requirement just stated is then translated into the k-anonymity requirement below, which states that every tuple released cannot be related to fewer than k respondents.

Definition 1 (k-anonymous requirement): Each release of data must be such that every combination of values of quasi-identifiers can be indistinguishably matched to at least k respondents.

Since it seems impossible or highly impractical to make assumptions on the datasets available for linking to external attackers or curious data recipients, essentially k-anonymity takes a safe approach requiring that the respondents should be indistinguishable (within a given set) with respect to the set of attributes in the released table. To guarantee the k-anonymity requirement, k-anonymity requires each value of a quasi-identifier in the released table to have at least k occurrences. Formally, we have the following definition.

Definition 2 (k-anonymity): Let PT(A1, ..., Am) be a private table and QI be a quasi-identifier associated with it. PT is said to satisfy k-anonymity with respect to QI if and only if each sequence of values in PT[QI] appears at least with k occurrences in PT[QI].2

If a set of attributes of external tables appears in the quasi-identifier associated with the private table (PT) and the table satisfies k-anonymity, then the combination of the released data with the external data will never allow the recipient to associate each released tuple with less than k respondents. For instance, consider the released microdata in Table 1 with quasi-identifier QI = {Gender, Age, Zip}, we see that the table satisfies k-anonymous with k = 1 only since there exists single occurrence of values over the considered QI (e.g., the single occurrence of “Male, 22 and 4352”).

3 Hash table

A hash table is a data structure that will increase the search efficiency from O(\log(n)) (binary search) to O(1) (constant time) (Cormen et al. 2001). A hash table is made up of two parts: an array (the actual table where the data to be searched is stored) and a mapping function, known as a hash function. The hash function is a mapping from the input data space to the integer space that defines the indices of the array (bucket). In other words, the hash function provides a way for assigning numbers to the input data such that the data can then be stored at the array (bucket) with the index corresponding to the assigned number. For example, the data in Table 1 are mapped into buckets labeled 0, 1, 2, 3 in Table 2. The data in the bucket with the same assigned number is called a hash equivalence class. Depending on the different problems, we could choose different hash functions to classify our input data as we need. For instance, consider quasi-identifier QI = {Age, Zip} in Table 1. We hash them into different buckets with the function ((Age − 20) + (Zip − 4350)) mod 4 (see Table 2).

From Table 2 we see that two identical data (25, 4350) and (28, 4373) in the quasi-identifier fall into two different hash equivalence classes. Further, if we add a row (labeled COUNT) to record the number of contents in the corresponding bucket (see Table 3), we can easily determine whether or not the table satisfies the k-anonymity requirement. For instance, according to the row COUNT in Table 3, Table I only satisfies k-anonymity with k = 1.

This hash-based technique is not new in data mining. In (Park et al. 1995), the authors used this technique to present an efficient hash-based algorithm for mining association rules which improves previous well-known A priori algorithm. In this paper, we integrate this technique into computation of minimal k-anonymous table. By using such a technique, we can reduce the number of potential sets that need to be checked whether they are k-anonymous during binary search and thus improve the time complexity in (Samarati 2001).

Concerning the efficiency of hash table and binary search, we note the following. (1) Hash table has a faster average lookup time O(1) (Cormen et al. 2001)3 than the binary search algorithm O(\log(n)). Hash

### Table 1: Released microdata

<table>
<thead>
<tr>
<th>Gender</th>
<th>Age</th>
<th>Zip</th>
<th>Other Attributes</th>
<th>Diseases</th>
</tr>
</thead>
<tbody>
<tr>
<td>Male</td>
<td>25</td>
<td>4370</td>
<td>··· ···</td>
<td>Hypertension</td>
</tr>
<tr>
<td>Male</td>
<td>25</td>
<td>4370</td>
<td>··· ···</td>
<td>Hypertension</td>
</tr>
<tr>
<td>Male</td>
<td>22</td>
<td>4352</td>
<td>··· ···</td>
<td>Depression</td>
</tr>
<tr>
<td>Female</td>
<td>28</td>
<td>4374</td>
<td>··· ···</td>
<td>Chest Pain</td>
</tr>
<tr>
<td>Female</td>
<td>28</td>
<td>4374</td>
<td>··· ···</td>
<td>Obesity</td>
</tr>
<tr>
<td>Female</td>
<td>34</td>
<td>4356</td>
<td>··· ···</td>
<td>Flu</td>
</tr>
</tbody>
</table>

### Table 2: Hashed table 1 with QI = (Age, Zip)

<table>
<thead>
<tr>
<th>Bucket</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>(22, 4352)</td>
<td>(25, 4370)</td>
<td>(34, 4350)</td>
<td>(28, 4373)</td>
</tr>
</tbody>
</table>

### Table 3: Hash table with COUNT

<table>
<thead>
<tr>
<th>Bucket</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>COUNT</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

2PT[QI] denotes the projection, maintaining duplicate tuples, of attributes QI in PT.

3Note that the worst case in hash tables happens when every data element are hashed to the same value due to some bad luck in choosing the hash function and bad programming. In that case, to do a lookup, we would really be doing a straight linear search on a linked list, which means that our search operation is back to being O(n). The worst case search time for a hash table is O(n). However, the probability of that happening is so small that, while the worst case search time is O(n), both the best and average cases are O(1).
table shines in very large arrays, where \( O(1) \) performance is important. (2) Building a hash table requires a reasonable hash function, which sometimes can be difficult to write well, while binary search requires a total ordering on the input data. On the other hand, with hash tables the data may be only partially ordered.

4 Data generalization

4.1 Generalization relationship

Among the techniques proposed for providing anonymity in the release of microdata, the \( k \)-anonymity proposal focuses on two techniques in particular: generalization and suppression, which unlike other existing techniques, such as scrambling or swapping, preserve the truthfulness of the information.

Generalization consists in substituting the values of a given attribute with more general values. We use \( \ast \) to denote the more general value. For instance, we could generalize two different Zip code 4370 and 4373 to 43\( \ast \). The other technique, referred to as data suppression, removes the part or entire value of attributes from the table. Since suppressing an attribute (i.e., not releasing any of its values) to reach \( k \)-anonymity can equivalently be modeled via a generalization of all the attribute values to the most generalized data \( \ast \), we consider only data generalization.

The notion of domain (i.e., the set of values that an attribute can assume) is extended to capture the generalization process by assuming the existence of a set of generalized domains. The set of original domains together with their generalizations is referred to as \( \text{Dom} \). Each generalized domain contains generalized values and there exists a mapping between each domain and its generalizations. (For example, Zip codes can be generalized by dropping the least significant digit at each generalization step, Ages can be generalized to an interval, and so on.) This mapping is described by means of a generalization relationship \( \leq_D \). Given two domains \( D_i \) and \( D_j \) in \( \text{Dom} \), \( D_i \leq_D D_j \) states that values in domain \( D_j \) are generalizations of values in \( D_i \). The generalization relationship \( \leq_D \) defines a partial order on the set \( \text{Dom} \) of domains, and is required to satisfy the following two conditions:

\[ C_1: \forall D_i, D_j, D_k \in \text{Dom}: \]
\[ D_i \leq_D D_j, D_j \leq_D D_k \Rightarrow D_i \leq_D D_k \]
\[ C_2: \text{all maximal element of } \text{Dom} \text{ are singleton.} \]

Condition \( C_1 \) states that for each domain \( D_i \), the set of domains generalization of \( D_i \) is totally ordered and we can think of the whole generalization domain as a chain of nodes, and if there is an edge from \( D_i \) to \( D_j \), we call \( D_j \) the direct generalization of \( D_i \). Note that the generalization relationship \( \leq_D \) is transitive, and thus, if \( D_i \leq_D D_j \) and \( D_j \leq_D D_k \), then \( D_i \leq_D D_k \). In this case, we call \( D_j \) the implied generalization of \( D_i \). Condition \( C_1 \) implies that each \( D_i \) has at most one direct generalization domain \( D_j \), thus ensuring determinism in the generalization process. Condition \( C_2 \) ensures that all values in each domain can be generalized to a single value. For each domain \( D \in \text{Dom} \), the definition of a generalization relationship implies the existence of a totally ordered hierarchy, called the domain generalization hierarchy, denoted \( DGH_D \). Paths in the domain generalization hierarchy correspond to implied generalizations and edges correspond to direct generalizations. For example, consider \( DGH_Z \) in Figure 1. \( Z_1 \) is the direct generalization of \( Z_0 \) and \( Z_2 \) is the implied generalization of \( Z_0 \).

A value generalization relationship denoted \( \leq_V \), can also be defined, which associates with each value in domain \( D_i \) a unique value in domain \( D_j \). For each domain \( D \in \text{Dom} \), the value generalization relationship implies the existence of a value generalization hierarchy, denoted \( VGH_D \). It is easy to see that the value generalization hierarchy \( VGH_D \) is a tree, where the leaves are the minimal values in \( D \) and the root (i.e., the most general value) is the value of the maximum element in \( DGH_D \).

EXAMPLE: Figure 1 illustrates an example of domain and value generalization hierarchies for domains: \( Z_0, A_0 \) and \( G_0 \). \( Z_0 \) represents a subset of the Zip codes in Table 1: \( A_0 \) represents Age; and \( G_0 \) represents Gender. The generalization relationship specified for Zip codes generalizes a 4-digit Zip code, first to a 3-digit Zip code, and then to a 2-digit Zip code. The attribute Age is first generalized to the interval (22-25) and (28-34), then to the interval (22-34). The Gender hierarchy in the figure is of immediate interpretation.

Since the approach in (Samarati 2001) works on sets of attributes, the generalization relationship and hierarchies are extended to refer to tuples composed of elements of \( \text{Dom} \) and/or of their values. Given a domain tuple \( DT = (D_1, \ldots, D_n) \) such that \( D_i \in \text{Dom} \), \( i = 1, \ldots, n \), the domain generalization hierarchy of \( DT \) is \( DGH_{DT} = DGH_{D_1} \times \cdots \times DGH_{D_n} \), where the Cartesian product is ordered by imposing coordinate-wise order. Since each \( DGH_{D_i} \) is totally ordered, \( DGH_{DT} \) defines a lattice with \( DT \) as its minimal element and the tuple composed of the top of each \( DGH_{D_i}, i = 1, \ldots, n \) as its maximal element. Each path from \( DT \) to the unique maximal element of \( DGH_{DT} \) defines a possible alternative path, called generalization strategy for \( DGH_{DT} \), which can be followed when generalizing a quasi-identifier \( QI = (A_1, \ldots, A_m) \) of attributes on domains \( D_1, \ldots, D_n \). In correspondence with each generalization strategy of a domain tuple, there is a value generalization strategy describing the generalization at the value level. Such a generalization strategy hierarchy is actually a tree structure. The top unique maximal element can be regarded as the root of the tree and the minimal element on the bottom is the leaf of the tree. Let \( L[i, j] \) denote the \( j^{th} \) data at height \( i \). The bottom data is at the height 0 and \( L[i] \) denote the number of data at the height \( i \).

EXAMPLE: Consider domains \( G_0 \) (Gender) and \( Z_0 \) (Zip code) whose generalization hierarchies are illustrated in Figure 1. Figure 2 illustrates the domain generalization hierarchy of the domain tuple \( < G_0, Z_0 > \) together with the corresponding do-
main and value generalization strategies. There are three different generalization strategies corresponding to the three paths from the bottom to the top element of lattice $DGH_{<G_0,Z_0>}$. In the generalization strategy 1, $L[0,2]$ is $\{\text{male}, 4370\}$, $L[0] = 6$ and $L[2,2]$ is $\{\text{person}, 435*\}$, $L[2] = 2$.

### 4.2 Generalized table and minimal generalization

Given a private table ($PT$), our approach to provide $k$-anonymity is to generalize the values stored in the table. Intuitively, attribute values stored in the private table ($PT$) can be substituted with generalized values upon release. Since multiple values can be mapped to a single generalized value, generalization may decrease the number of distinct tuples, thereby possibly increasing the size of the clusters containing tuples with the same values. We perform generalization at the attribute level. Generalization at the attribute level ensures that all values of an attribute belong to the same domain. In the following, $dom(A_i, PT)$ denotes the domain of attribute $A_i$ in private table $PT$.

**Definition 3 (Generalized table):** Let $PT(A_1, \ldots, A_n)$ and $PT(A_1, \ldots, A_n)$ be two tables defined in the same set of attributes. $PT_i$ is said to be a generalization of $PT_i$, written $PT_i \preceq PT_i$, if and only if: (1) $|PT_i| = |PT_j|$; (2) $\forall A_k \in \{A_1, \ldots, A_n\}$ : $dom(A_k, PT_i) \subseteq_D dom(A_k, PT_j)$; and (3) It is possible to define a bijection mapping between $PT_i$ and $PT_j$ that associates each tuple $pt_i \in PT_i$ with a tuple $pt_j \in PT_j$ such that $pt_i[A_k] \leq V pt_j[A_k]$ for all $A_k \in \{A_1, \ldots, A_n\}$.

**EXAMPLE:** Consider the private table $PT$ illustrated in Figure 3.1 and the domain and value generalization hierarchies for $G_0(Gender)$ and $Z_0(\text{Zip})$ illustrated in Figure 2. Assume $QI = \{\text{Gender}, \text{Zip}\}$ to be a quasi-identifier. The following three tables in Figure 3 are all possible generalized tables for $PT$. For the clarity, each table reports the domain for each attribute in the table. With respect to $k$-anonymity, $GT_{[1,1]}$ satisfies $k$-anonymity for $k = 1, 2$; $GT_{[0,2]}$ satisfies $k$-anonymity for $k = 1, 2, 3$ and $GT_{[1,2]}$ satisfies $k$-anonymity for $k = 1, \ldots, 6$.

Given a private table $PT$, different possible generalizations exist. However, not all generalizations can be considered equally satisfactory. For instance, the trivial generalization bringing each attribute to the highest possible level of generalization provides $k$-anonymity at the price of a strong generalization of the data. Such extreme generalization is not needed if a table containing more specific values exists which satisfies $k$-anonymity as well. This concept is captured by the definition of minimal $k$-anonymity (generalization). To introduce it we first introduce the notion of distance vector.\(^5\)

**Definition 4 (Distance vector):** Let $PT(A_1, \ldots, A_n)$ and $PT(A_1, \ldots, A_n)$ be two tables such that $PT_i \preceq PT_j$. The distance vector of $PT_j$ from $PT_i$ is the vector $DV_{ij} = [d_1, \ldots, d_n]$ where each $d_z = 1$ if $\forall A_k \in \{A_1, \ldots, A_n\}$ such that $dom(A_k, PT_i) \subseteq_D dom(A_k, PT_j)$ in the domain generalization hierarchy $DGH_D$.

**EXAMPLE:** Consider the private table $PT$ and its generalizations illustrated in Figure 3. The distance vectors between $PT$ and each of its generalized tables is the vector appearing as a subscript of the table. A generalization hierarchy for a domain tuple can be seen as a hierarchy (lattice) on the corresponding distance vectors. Figure 4 illustrates the lattice

\(^5\)In (LeFevre et al. 2005) the star scheme is used for large databases. Here, we use distance vector to define minimal $k$-anonymity.
representing the dominance relationship between the distance vectors corresponding to the possible generalizations of \(<G,D,Y>_0>\).

We extend the dominance relationship \(\leq_{D}\) on integers to distance vectors by requiring coordinate-wise ordering as follows. Given two distance vectors \(DV = [d_1, \ldots, d_n]\) and \(DV' = [d'_1, \ldots, d'_n]\), \(DV \leq DV'\) if and only if \(d_i \leq d'_i\) for all \(i = 1, \ldots, n\). Moreover, \(DV < DV'\) if and only if \(DV \leq DV'\) and \(DV \neq DV'\).

Intuitively, a generalization \(PT_i(A_1, \ldots, A_n)\) is minimal \(k\)-anonymity (generalization) if and only if there does not exist another generalization \(PT_i(A_1, \ldots, A_n)\) satisfying \(k\)-anonymity and whose domain tuple is dominated by \(PT_i\) in the corresponding lattice of distance vectors. Formally, we can define it as follows:

**Definition 5 (Minimal \(k\)-anonymity):** Let \(PT_i(A_1, \ldots, A_n)\) and \(PT_j(A_1, \ldots, A_n)\) be two tables such that \(PT_i \leq PT_j\). \(PT_j\) is said to be a minimal \(k\)-anonymity (generalization) of \(PT_i\) if and only if:

1. \(PT_i\) satisfies \(k\)-anonymity; and
2. \(\forall PT' \leq PT_i\): \(PT_j \leq PT'\) satisfies \(k\)-anonymity \(\Rightarrow (DV_{i,z} \leq DV_{i,j})\).

**EXAMPLE:** Consider table \(PT\) and its generalized tables illustrated in Figure 3. For \(k = 2\) two minimal \(k\)-anonymous table exist, namely \(GT_{[0,0]}\) and \(GT_{[1,1]}\). \(GT_{[1,2]}\) is not minimal because it is a generalization of \(GT_{[1,1]}\) and \(GT_{[0,2]}\). Also, there is only one minimal \(k\)-generalized tables with \(k = 3\), which is \(GT_{[0,2]}\).

### 5 Hash-based algorithm

A number of convincing parallels exist between Samarati and Sweeney’s generalization framework (Samarati & Sweeney 1998, Samarati 2001) and ideas used in mining association rules (Agrawal & Srikant 1994, Srikant & Agrawal 1995) and the hash-based technique used in (Park et al. 1995). By bringing these techniques to bear on our model of full-domain generalization problem, we develop an efficient hash-based algorithm for computing \(k\)-minimal anonymity.

In (Samarati 2001), Samarati describes an algorithm for finding a single minimal \(k\)-anonymous full-domain generalization based on the specific definition of minimality outlined in the previous section. The algorithm uses the observation that if no generalization of height \(h\) satisfies \(k\)-anonymity, then no generalization of height \(h' < h\) will satisfy \(k\)-anonymity. For this reason, the algorithm performs a binary search on the height value. If the maximum height in the generalization lattice is \(h\), the algorithm begins by checking each generalization at height \(\lceil \frac{h}{2} \rceil\). If a generalization exists at this height that satisfies \(k\)-anonymity, the search proceeds to look at the generalizations of height \([ \frac{h}{2} \rceil \) and higher. Otherwise, generalizations of height \([ \frac{3h}{4} \rceil \) are searched, and so forth. This algorithm is proven to find a single minimal \(k\)-anonymous table.

We integrate the hash technique into the algorithm and develop a more efficient algorithm based on our definition of minimality (Definition 5). A drawback of Samarati’s algorithm is that for arbitrary definitions of minimality this binary search algorithm is not always guaranteed to find the minimal \(k\)-anonymity table. We conjecture that the hash technique used in this paper might be suitable for the further improvement of algorithms based on other optimal criteria for \(k\)-anonymity.

Let the domain generalization hierarchy be \(DGH_{DT}\), where \(DT\) is the tuples of the domains of the quasi-identifier. Assume that the top generalization data with the highest height in \(DGH_{DT}\) satisfies the required \(k\)-anonymity. The idea of the algorithm is to hash the data in \(DGH_{DT}\) to a different hash equivalence class. Under our definition of the minimality, the hash function that we choose should hash all generalizations with height \(h > 0\) in \(DGH_{DT}\) that satisfies \(k\)-anonymity to the same hash equivalence class, which is called the \(k\)-anonymous class. (The bucket labeled 2 in Table 4). The hash-based algorithm consists of two main steps. At the first stage, the data that satisfies \(k\)-anonymity are hashed into the \(k\)-anonymous class. The second stage is to use Algorithm 1 to find the minimal \(k\)-anonymous table in the \(k\)-anonymous class.

Algorithm 1 illustrate how to find the minimal \(k\)-anonymous table in \(k\)-anonymous class. Consider Table 1 and its Generalization Strategy 1 in Figure 2. Generalized data \(L[1,1], L[1,2], L[2,1], L[2,2]\) and \(L[3,1]\) are hashed into the \(k\)-anonymous class. We sort the data in \(k\)-anonymous class as \(\{L[1,1], L[1,2], L[2,1], L[2,2], L[3,1]\}\). Since \(L[1] = 4\) and the number of data at the height 1 in \(k\)-anonymous class is 2. According to Step 3 in Al-

<table>
<thead>
<tr>
<th>Bucket</th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>([0,1], [0,2], [0,3])</td>
<td>([1,3])</td>
<td>([1,1], [1,2])</td>
</tr>
<tr>
<td></td>
<td>([0,4], [0,5], [0,6])</td>
<td>([1,4])</td>
<td>([2,1], [2,2], [3,1])</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 1: Finding minimal solution in (k)-anonymous class.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> the (k)-anonymous class</td>
</tr>
<tr>
<td>1. Sort the data in (k)-anonymous class.</td>
</tr>
<tr>
<td>2. Compute the number (n(i)) of ([l,j]) at each height (i).</td>
</tr>
<tr>
<td>3. If (n(i) \neq L[i,j]) discard all the ([l,j]) at the height (i)</td>
</tr>
<tr>
<td>4. Otherwise, keep them.</td>
</tr>
<tr>
<td><strong>Output:</strong> The height at which the first data is in the remaining (k)-anonymous class, and generalize the data to this height could obtain the minimal (k)-anonymous table.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Algorithm 2: Hash-based algorithm for minimal (k)-anonymity.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> Generalization hierarchy and anonymous requirement (k)</td>
</tr>
<tr>
<td><strong>Output:</strong> A minimal (k)-anonymous table.</td>
</tr>
</tbody>
</table>
| 1. Create a table with \(k+1\) column labeling \(0, 1, \ldots, k\).
| Compute \(Children[i,j]\) for each data \(j\) at the height \(i\).
| 2. For \(l = 0, 1, \ldots, k-1\)
| if \(Children[i,j] = l\), put \(Children[i,j]\) to the bucket labeled \(l\).
| else put \(Children[i,j]\) to the bucket labeled \(k\).
| 3. Compute the minimal \(k\)-anonymous table by Algorithm 1. |

---

Table 4: Hash table of Generalization Strategy 1 in Figure 2.

---

Figure 2: Generalized data \(DGH_{c_{0}, z_{0}}\) and corresponding lattice on distance vectors.

Figure 4: Hierarchy \(DGH_{c_{0}, z_{0}}\) and corresponding lattice on distance vectors.
Algorithm 1, we delete \( L[1,1] \) and \( L[1,2] \) from \( k \)-anonymous class. At last, the output height is 2, and we could generalize the table to this height so that it satisfies 2-anonymity with quasi-identifier \( QI = \{ \text{Gender}, \text{Zip} \} \).

Next, we illustrate how to hash the generalization data in \( DGH_{DT} \) to the \( k \)-anonymous class. Denote \( \text{Children}[i,j] \) the number of children that the \( j^{th} \) data at the height \( i \) have. For example, in Generalization Strategy 1 in Figure 2, \( \text{Children}[1,3] = 1 \) and \( \text{Children}[2,1] = 4 \). Suppose we have the requirement of \( k \)-anonymity. The desired hash table contains \( k + 1 \) buckets, labeled as \( 0, 1, 2, \ldots, k-1, k \), the labeled number 0, 1, \ldots, \( k-1 \) denotes the value of \( \text{Children}[i,j] \) in \( DGH_{DT} \) and the \( k^{th} \) bucket has the data whose \( \text{Children}[i,j] \geq k \). Note that the bucket labeled \( k \) is actually the \( k \)-anonymous class. We could see the following Table 4 as an example (where \( k = 2 \)). All the potential generalization data satisfying \( 2 \)-anonymity are classified into the third bucket, which consists of the \( k \)-anonymous class.

Algorithm 2 is our hash-based algorithm. Compared to Samarati’s binary search algorithm, Algorithm 2 finds the minimal \( k \)-anonymous table in the \( k \)-anonymous class, which is smaller than the potential sets that need to be checked in Samarati’s algorithm. Because of the hash technique we used in Algorithm 2, the search complexity is reduced from \( O(\log(n)) \) (binary search) to \( O(1) \) (Cormen et al. 2001).

6 Related work

Protecting anonymity when publishing microdata has long been recognized as a problem (Willenborg & De-Waal 2001), and there has been much recent work on computing \( k \)-anonymity for this purpose. The \( \mu \)-Argus system (Hundepool & Willenborg 1996) was implemented to anonymize microdata but considered attribute combinations of only a limited size, so the results were not always guaranteed to be \( k \)-anonymous.

In recent years, numerous algorithms have been proposed for implementing \( k \)-anonymity via generalization and suppression. The framework was originally defined by Samarati and Sweeney (Samarati & Sweeney 1998). Sweeney proposed a greedy heuristic algorithm for full-domain generalization ("Datafly") (Sweeney 2002). Although the resulting generalization is guaranteed to be \( k \)-anonymous, there are no minimality guarantees. Samarati proposed the binary search algorithm for discovering a single minimal full-domain generalization that is described in Section 5. LeFevre et al. described an efficient search algorithm called Incognito, for anonymous full-domain generalization (LeFevre et al. 2005).

Cost metrics intended to quantify loss of information due to generalization were described in (Iyengar 2002). Given such a cost metric, Iyengar (Iyengar 2002) developed a genetic algorithm and Winkler (Winkler 2002) described a stochastic algorithm based on simulated annealing to find locally minimal anonymous table. Recently, top-down (Fung et al. 2005) and bottom-up (Wang, Yu & Chakraborty 2004) greedy heuristic algorithms were proposed to produce anonymous data.

Bayardo and Agrawal (Bayardo & Agrawal 2005) described a set enumeration approach to find an optimal anonymous table according to a given cost metric. Subsequent work shows that optimal anonymity under this model may not be as good as anonymity produced with a multi-dimension variation (LeFevre et al. 2005). Finally, Meyerson and Williams (Meyerson & Williams 2004) and Aggarwal et al. (Aggarwal et al. 2005) proved the optimal \( k \)-anonymity is \( NP \)-hard (based on the number of cells and number of attributes that are generalized and suppressed) and describes approximation algorithms for optimal \( k \)-anonymity.

In addition to generalization and suppression, related techniques based on clustering have also been proposed in the literature. Microaggregation first clusters data into (ideally homogeneous) groups of required minimal occupancy and then publishes the centroid of each group (Domingo-Ferrer & Mateos-Sanz 2002). Similarly, Aggarwal et al. propose clustering data into groups of at least size \( k \) and then publish various summary statistics for each cluster (Aggarwal et al. 2006).

7 Conclusion and future work

In this paper, we focus on a specific global-recoding model of \( k \)-anonymity. Our objective is to find the minimal \( k \)-anonymous generalization (table) under the definition of minimality defined by Samarati (Samarati 2001). By introducing the hash-based technique, we provide a new approach to generate minimal \( k \)-anonymous table, which not only improves previous search algorithm proposed by Samarati (Samarati 2001), but might be useful for computing other optimal criteria solution for \( k \)-anonymity.

In future work, we conjecture this hash-based technique might be suitable for further improvement of the Incognito (LeFevre et al. 2005) for full-domain generalization, since it might significantly reduce the number of \( 2 \)-attribute candidate sets. The technique might also apply in multilevel generalization. For many applications, it is difficult to find required \( k \)-anonymity tables at low or primitive levels of the generalization hierarchy due to the sparsity of data in multidimensional space. \( k \)-anonymous table generated at very high levels in the generalization hierarchy might be suitable for some attributes, however, to some extent, it may be too generalized in some attributes. Therefore, data mining system should provide capability to generate \( k \)-anonymous tables at multiple levels of the generalization hierarchy and traverse easily among different generalization levels. The hash-based technique may provide a new point of view and a more efficient way to make multilevel \( k \)-anonymous tables.

References


Bayardo, R. and Agrawal, R. Data privacy through optimal \( k \)-anonymity. In Proceedings of the 21st
International Conference on Data Engineering (ICDE), 2005.


Samarati P. Protecting respondents’ identities in microdata release. IEEE Transactions on Knowledge and Data Engineering, 13(6):1010-1027. 2001


