PrivacyFrost2: A Efficient Data Anonymization Tool Based on Scoring Functions

Shinsaku Kiyomoto, Yutaka Miyake

PrivacyFrost2: A Efficient Data Anonymization Tool Based on Scoring Functions

Shinsaku Kiyomoto and Yutaka Miyake

KDDI R & D Laboratories Inc.
2-1-15 Ohara, Fujimino-shi, Saitama, 356-8502, Japan
kiyomoto@kddilabs.jp

Abstract. In this paper, we propose an anonymization scheme for generating a k-anonymous and l-diverse (or t-close) table, which uses three scoring functions, and we show the evaluation results for two different data sets. Our scheme is based on both top-down and bottom-up approaches for full-domain and partial-domain generalization, and the three different scoring functions automatically incorporate the requirements into the generated table. The generated table meets users' requirements and can be employed in services provided by users without any modification or evaluation.

1 Introduction

Anonymization methods have been considered as a possible solution for protecting private information[8]. One class of models, called *global-recoding*, maps the values of attributes to other values [29] in order to generate an anonymized dataset. This paper uses a specific global-recoding model, "*full-domain generalization*", and an additional process for local optimization. Generally, some anonymous tables are generated from an original table and users select a table from these based on certain requirements for the services that they provide to the public. A challenging issue in the anonymization of tables is to realize an algorithm that generalizes a table according to the requirements of a data user. If the algorithm incorporates the requirements into a generated table and outputs the most suitable table, then evaluation and selection of candidates for anonymous tables are not required when using an anonymous table.

In this paper, we propose an anonymization mechanism that reflects the user's requirements. The mechanism is an extension of *PrivacyFrost* [12, 11], and hold the same properties that are summarized as follows:

- The mechanism generates an anonymous table that satisfies k-anonymity[22], l-diversity-family (l-diversity[18] and recursive (c, l)-diversity[18]).
- The mechanism is constructed based on a combination of top-down and bottom-up methods for full-domain generalization, and produces the anonymous table that has the best score in the execution.
- After full-domain generalization, the mechanism executes the top-down algorithm on each segment to optimize the anonymous table.

– A user inputs not only a set of generalization boundary constraints, but also priorities for quasi-identifiers as the user's requirements for the anonymous table.

We improve the tool to generate more valuable data. Extensions presented in this paper are:

– A privacy notion, t-closeness [19], is supported for anonymization, and we compare the transaction time with that of k-anonymity and l-diversity-family cases.
– Three different scoring functions can be selected for anonymization. The three scoring functions outputs anonymous tables that have characteristic properties for the selected scoring function; thus, it can be selected according to a requirement for each anonymous table.
– A pre-sampling process removes *isolated* records in order to output a more useful table. To optimize output data, we can use a pre-sampling process before execution of the algorithm.

It realizes fast generation of an anonymous table to reflect a user's requirements and reduce the number of candidates for an anonymous table. The mechanism evaluates the score of a table for each iteration of generalization and selects the best scoring table for the next iteration. After some iterations, the mechanism provides a k-anonymous and l-diverse (or t-close) table that is suited to the user's requirements.

The rest of the paper is organized as follows: Related work is presented in Section 2. Section 3 presents assumed requirements from data users. Our mechanism is presented in Sections 4, 5, and 6. We show evaluation results in Section 7 and conclude this paper in Section 8.

2 Related Work

Samarati and Sweeney [23, 22, 26] proposed a primary definition of privacy that is applicable to generalization methods. A data set is said to have *k-anonymity* if each record is indistinguishable from at least $k-1$ other records with respect to certain identifying attributes called *quasi-identifiers* [9]. In other words, at least k records must exist in the data set for each combination of the identifying attributes. Clearly any generalization algorithm that converts a database into one with k-anonymity involves a loss of information in that database.

Minimizing this information loss thus presents a challenging problem in the design of generalization algorithms. The optimization problem is referred to as the k-anonymity problem. Meyerson reported that optimal generalization in this regard is an NP-hard problem[20]. Aggarwal *et al.* proved that finding an optimal table including more than three attributes is NP-hard [2]. Nonetheless, k-anonymity has been widely studied because of its conceptual simplicity [4, 18, 19, 30, 28, 25]. Machanavajjhala *et al.* proposed another important definition of privacy for a public database [18]. The definition, called l-diversity, assumes a

strong adversary having certain background knowledge that allows the adversary to identify object persons in the public database.

Samarati proposed a simple binary search algorithm for finding a k-anonymous table[22]. A drawback of Samarati's algorithm is that for arbitrary definitions of minimality, it is not always guaranteed that this binary search algorithm can find the minimal k-anonymity table. Sun *et. al.* presented a hash-based algorithm that improves the search algorithm[24]. Aggarwal *et al.* proposed the $O(k)$-approximation algorithm [3] that is used for executing the k-anonymity problem. A greedy approximation algorithm [14] proposed by LeFevre *et al.* searches for the optimal multi-dimensional anonymization. A genetic algorithm framework [10] was proposed because of its flexible formulation and its ability to find more efficient anonymizations. Utility-based anonymization [33, 32] makes k-anonymous tables using a heuristic local recoding anonymization. Moreover, the k-anonymization problem is viewed as a clustering problem. Clustering-based approaches [7, 27, 16, 34] search a cluster that has k-records. In full-domain generalization, there are two heuristic approaches for generalization algorithms: the top-down approach and the bottom-up approach. Bayardo and Aggrawal proposed a generalization algorithm using the top-down approach [6]. The algorithm finds a generalization that is optimal according to a given fixed cost metric for a systematic search strategy, given generalization hierarchies for a single attribute. Incognito [13] is a bottom-up-based algorithm that produces all possible k-anonymous tables from an original table.

There are several research papers about k-anonymization based on a data owner's requirements for anonymized public data. Loukides *et al.* considered a k-anonymization approach [17] according to both the data owner's policies and data user's requirements. Aggarwal and Yu discussed a condensation based approach [1] for different privacy requirements. LeFevre *et al.* provides an anonymization algorithm [15] that produces an anonymous view appropriate for specific data mining tasks. Xiao and Tao proposed the concept of *personalized anonymity* and presented a generalization method [31] that performs the minimum generalization for the requirements of data owners. Miller *et al.* presented an anonymization mechanism [21] that provides k-anonymous tables under generalization boundaries of quasi-identifiers. The configuration of the boundaries can be considered to be user requirements. We proposed an anonymization mechanism[12, 11] to reflect user requirements precisely. However, the mechanism only provides a basic scoring function for the anonymization and a remaining issue is to design appropriate scoring functions for achieving several requirements on anonymous tables. This paper presents an extension of that mechanism that includes presampling and some scoring functions.

3 Requirements

A database table T in which the attributes of each user are denoted in one record is in the public domain and an attacker obtains the table and tries to extract the record of an individual. Suppose that a database table T has m records and

n attributes $\{A_1, \ldots, A_n\}$. Each record $a^i = (a^i_1, \ldots, a^i_n)$ can thus be considered to be an n-tuple of attribute values, where a^i_j is the value of attribute A_j in record a^i. The database table T itself can thus be regarded as the set of records $T = \{\mathbf{a^i} : 1 \leq i \leq m\}$. In our system, a user can input the following conditions for the anonymization.

- *Priority.* The user defines a positive integer value v_{a^i} for each attribute a^i. The value depends on the priority of the attribute. That is, $v_{a^i} = mv_{a^j}$, where the priority of the attribute a^i is m-times higher than a^j. For example, the user can define $(v_{a^1}, v_{a^2}, v_{a^3}, v_{a^4}) = (10, 5, 1, 1)$. The user gives high priority to an attribute when the user desires more detailed information about the attribute.
- *Minimum Level.* The user can define a minimum level for each attribute. Each attribute has a hierarchical tree structure. The minimum level $w^M_{a^i}$ defined by the user means that a k-anonymized dataset generated by the system includes at least $w^M_{a^i}$-level information for the attribute. The system does not generalize the attribute below the minimum level.

The above two requirements reflect the granularity of information in a generated anonymous table. The mechanism tries to keep attribute values located at the lower node as much as possible in the generalization hierarchy while satisfying the predefined anonymity condition, when the user marks the attribute as high priority. Furthermore, the user controls the limits of generalization using a configuration of minimum levels for attributes.

4 Components

In this section, we explain the basic components of our anonymization method.

4.1 Generalization

Full-domain generalization for obtaining an anonymous table consists of replacing attribute values with a generalized version of those values, and it is based on generalization hierarchies[8]. A quasi-identifier is an attribute that can be joined with external information to re-identify individual records with sufficiently high probability [9]. Generally, a target table $T^x = (T^q|T^s)$ consists of two types of information: a subtable of quasi-identifiers T^q and a subtable of sensitive attributes T^s. Since the sensitive attributes represent the essential information with regard to database queries, a generalization method is used to modify (anonymize) T^q in order to prevent the identification of the owners of the sensitive attributes, while retaining the full information in T^s. Thus, the generalization algorithm focuses on the subtable of quasi-identifier T^q and modifies it to satisfy a predefined anonymity condition. We assume that quasi-identifier attributes are known information for the generalization algorithm.

The full-domain generalization modifies all values of the attribute in the table T. Therefore, the anonymized table is not optimized for regional groups

that have the same quasi-identifier T^q and that have lost information due to the global generalization of attributes. The partial-domain generalization replaces the values of an attribute in a small group that has the same quasi-identifier T_q. The partial-domain generalization executes [on] each group independently and modifies T_q subject to the requirement that each group satisfies the predefined anonymity condition. Thus, values of the attribute are generalized as values with different levels for each group.

4.2 Top-Down and Bottom-Up

There are two methods for generating anonymous tables in generalization schemes: the top-down approach and the bottom-up approach. The top-down approach starts at the root table where all attributes have a root value (a maximally generalized value), and finds an anonymous table to change attribute values to lower values (more detailed values) of the generalization hierarchy. In contrast, the initial table in the bottom-up approach is an original table and attribute values are replaced using upper attribute values until an anonymous table is found. Our scheme uses a top-down approach as the basic algorithm to reduce the number of score calculations. Furthermore, we consider a pre-computation in the top-down approach in order to skip some computations by starting at the root table. The details of our scheme are described in a later section.

4.3 Functions

The mechanism generates an anonymous table T^G and calculates its score s based on input data: a table $T = T^q$ that consists of m records and n quasi-identifiers a^i (i=1, ...,n), the parameters for k, a generalization hierarchy for the attributes H_{a^i}, the lowest levels $w^L_{a^i}$ of H_{a^i} and the user's requirements v_{a^i} and $w^M_{a^i}$. The parameters for k are the requirement for privacy (which means k-anonymity) and a system parameter that is defined according to the target table. The score s provides a rating of the degree to which the user's requirements are satisfied. The following subfunctions are used in the algorithm:

- Sort$(T, v_{a^1}, ..., v_{a^n})$. This function sorts attributes a^i $(i = 1, ..., n)$ in Table T by the user-defined priority values v_{a^i}. The function generates $T = (a^1, ..., a^n)$ in an order such that v_{a^1} is the smallest and v_{a^n} is the largest.
- Check$_{k,l,c,t}(T)$. This function calculates the minimum number of group members, where $T = (T_q|T_s)$ and all records in T_q are categorized into groups with the same attribute values. That is, this function calculates x of x-anonymity for T_q. The function calculates l and c of (c, l)-diversity for T_s, or calculates t for t-closeness for T_s. The function outputs OK where T satisfies k-anonymity, (c, l)-diversity, and otherwise outputs NG. The function skips calculations for l-diversity where l is defined as 1, and it only executes l-diversity calculations for (c, l)-diversity where c is defined as 0. The function skips calculations for t-closeness where $t = -1$. First, the function generates the hash value of each record in the table T_q and counts the number

of hash values that are the same. If all hash values are greater than x, the function outputs x. This process is a simple and efficient way for checking k-anonymity and is similar to the process adopted in the previous study. This function is implemented as a modifiable module; we can add other checking logics for anonymity definitions.
- Generalization(a^i, H_{a^i}, w_{a^i}). This function modifies the attribute a^i based on its generalization hierarchy H_{a^i}. The attribute values change to the upper node values of level w_{a^i-1}, where the level of the attribute value is w_{a^i}.
- De-Generalization(a^i, H_{a^i}, w_{a^i}). This function modifies the attribute a^i based on its generalization hierarchy H_{a^i}. The attribute values change to the lower node values of level w_{a^i+1}, where the level of the attribute value is w_{a^i}.
- Score(T). This function calculates the score of a table T. Our system calculates the score S_t of the anonymized datasets using equations described in Section 6.

5 Algorithm

In the generated table, information that is important for the user is maintained at a level that is as detailed as possible, while other information remains at the level of generalized information. The algorithm consists of two main parts: pre-computation and top-down generation.

5.1 Pre-Sampling

To optimize output data, we can use a pre-sampling process before execution of the algorithm. The pre-sampling process finds *isolated* records having attribute sets that very few records have, and removes the *isolated* records to output more useful tables. If the *isolated* records are included in a table, other records tend to be much generalized than expected, so deletion of the records helps to keep the maximum amount of information in the table. The pre-sampling process is an optional process that the user can choose to use, or not. Let $\mathbf{u} = u_{a^1}, ..., u_{a^n}$ be a threshold level of generalization for each attribute a^i, and k' be a threshold value of anonymization. When the user inputs a table T, and threshold values $\mathbf{u}$ and k', the pre-sampling process is executed as follows:

The algorithm uses the updated table T for generating the anonymous table T^G. The whole mechanism is described in Figure 1.

1. Each attribute a_i is generalized up to the level u_{a^i} and the process outputs a temporary generalized table T^P.
2. Using table T^P, the process makes groups whose records have the same attribute sets, and counts the number of records in each group. The process picks up records that belong to a group having records less than k'.
3. The selected records are removed from the original table T. The process outputs the updated table T. Note that the generalized table T^P is just discarded.

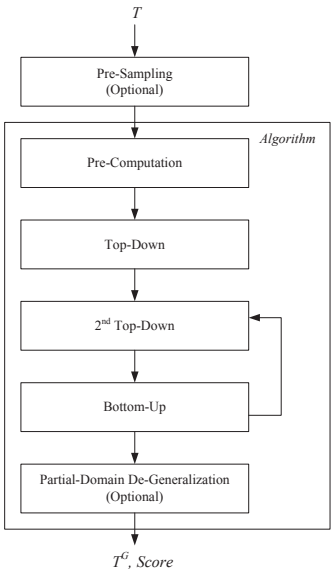

Fig. 1. Mechanism Overview

5.2 Pre-Computation

The mechanism uses pre-computation to reduce the total computational cost of anonymization. The pre-computation consists of two steps; step 1 considers single attribute anonymity and generalizes each attribute to satisfy $(k+p)$-anonymity, and step 2 considers the whole table to satisfy k-anonymity. The parameter p is a system parameter, and it should be optimized according to the results of previous trials. The pre-computation is based on the subset property theorem [13]. This theorem means that each single attribute has to satisfy k-anonymity for a k-anonymous table. The mechanism uses the top-down approach and starts with a k-anonymous table as the initial table; thus the algorithm executes the following pre-computation.

1. The algorithm generalizes each attribute of a table $T(= T^q)$ to satisfy $(k+p)$-anonymity, and creates a modified table T.
2. The algorithm checks whether the whole table T satisfies k-anonymity and (c, l)-diversity (or t-closeness). If the table T does not satisfy k-anonymity and (c, l)-diversity (or t-closeness), then the algorithm generalizes each attribute once (one level) and returns to step 2. Note that each attribute is not generalized up to the minimum level defined by the user. If the algorithm finds no table that satisfies k-anonymity and (c, l)-diversity (or t-closeness), then the algorithm outputs *failed*. Otherwise, the algorithm makes an initial anonymous table T^I and inputs it to T^G.

5.3 Top-Down Generalization

The basic steps of top-down generalization are as follows:

1. First, the algorithm de-generalizes an attribute a^n in Table T^G, which is defined as a top priority by the user, then checks the k-anonymity and (c, l)-diversity (or t-closeness) of the modified table T.
2. If the table satisfies the predefined anonymity condition, the algorithm calculates the score of the modified table T. If the algorithm finds no table that satisfies the predefined anonymity condition, then the algorithm outputs the table T^G and its score s. The score is computed using the scoring function Score(T).
3. For all possible modifications of a^i $(i = 1, ..., n)$, the algorithm checks k-anonymity and (c, l)-diversity (or$t$-closeness), then calculates the scores. The algorithm selects the table that has the largest score among the possibly anonymous tables and executes step 1 again.
4. If no more tables satisfy the predefined anonymity condition, the algorithm outputs the table that has the maximum score at that point.

The basic algorithm calculates the scores of all possible anonymous tables. Now, we consider a more efficient method for reducing the computational costs of anonymity checks and score calculations. Suppose that the algorithm obtains the number $b_{w_{a^i}}$ of nodes at the same level w_{a^i} in the generalization hierarchy H_{a^i}. The number indicates the upper bound of e_{a^i} in each level. Thus, we estimate the upper bound of increase in the score to calculate $\delta_s = (b_{w_{a^i}+1} - e_{a^i})v_{a^i}$, where the current generalization level of a^i is w_{a^i}. If $s > s' + \delta_s$, the algorithm can skip de-generalization of a^i. Figure 1 shows the basic algorithm that outputs T^G and s using input parameters. The algorithm first executes pre-computation and generates an initial table T^I; then the algorithm searches the table T^G that has the highest score. The top-down approach is used for the search performed by the basic algorithm.

5.4 Optimization Steps

We present optimization steps to find a better table that has a higher score than the table found using the basic algorithm. The optimization consists of three steps: the second top-down, bottom-up, and partial-domain de-generalization. The extension part is executed after the execution of the basic algorithm. The algorithm repeats two sub-functions until no table has a score higher than the current best score; then the algorithm executes the partial-domain generalization process, if required. An overview of the search process is shown in Figure 2. The figure is a generalization graph of the table T and the top is a root table in which all attributes are level 0. The extended algorithm tries to find a table using both top-down and bottom-up approaches. The algorithm searches for the high-score table on the boundary of anonymous tables and tables that do not satisfy the predefined anonymity condition. The extended algorithm is executed after the basic algorithm as follows:

1. After the basic algorithm, the algorithm generalizes possible a^is in the table T^G and calculates the scores of the tables. Then, the algorithm selects the top score table T^T. Note that table T^T does not satisfy $\text{Check}_{k,l,c}(T) = OK$.
2. Next, the algorithm executes a bottom-up generalization to search for tables that satisfy the predefined anonymity condition, then from these tables the algorithm chooses the table with the best score.
3. The algorithm compares the score of the chosen table with the current score s of table T^G, and if the score is higher than s, the algorithm replaces T^G with table T and s by its score. The algorithm executes the above steps using the new T^G. Otherwise, the algorithm stops and outputs the current generalized table T^G. The steps of top-down and bottom-up are repeated until no table has a score higher than the current best score.
4. After finding the best score table, the algorithm executes the top-down approach for each partial-domain using a function named "Partial-Domain De-Generalization (T)". The function selects one group that has the same quasi-identifiers and modifies one attribute using the top-down approach to satisfy the predefined anonymity condition. The algorithm executes the function until the table T^T does not satisfy $\text{Check}_{k,l,c}(T) = OK$. Note that the partial-domain generalization process consists of only the top-down approach, and the score of the output table is larger than that of the best scoring table produced by full-domain generalization. This step is optional; the user decides whether the partial-domain generalization process is executed for the table.

Details of the algorithm are shown in the Appendix.

6 Scoring Functions

In this section, we explain scoring functions that are used in the anonymization algorithm.

6.1 Basic Scoring Function

The basic function is described as follows:

$$S_t = \sum_{\forall a^i} v_{a^i} \cdot e_{a^i}$$

where e_{a^i} is the number of value types of an attribute a^i in the table. The score is high where the user-defined priority value for the attribute is high and the attribute has many types of values. The function produces a single value; thus, different priority values may produce the same value. The function is implemented as a replaceable module; thus, we can adjust the function or add another scoring function according to data types. Note that the table with the best score is not optimal among k-anonymous tables, but is suitable for the user's requirements.

Table by Logarithm-Based Scoring Function

Quasi-Identifiers			
Birth	Gender	Zip	Nationality
198*	*	012**	UK
198*	*	012**	Italy
198*	*	012**	UK
198*	*	012**	Italy
1984	Male	0123*	USA
1984	Male	0123*	USA

Table by η-Based Scoring Function

Quasi-Identifiers			
Birth	Gender	Zip	Nationality
1984	Male	0124*	Europe
1984	Male	0124*	Europe
1985	Female	0123*	Europe
1985	Female	0123*	Europe
1984	Male	0123*	USA
1984	Male	0123*	USA

Fig. 2. Generated Anonymous Tables

6.2 η-Based Scoring Function

The η-based scoring function focuses on the variation of attribute values for all attributes (calculated as the sum of e_{a^i}) more than in the case that we use the basic scoring function. Thus, the table tends to be uniformly generalized. The η-based scoring function is described as follows:

$$S_t = \sum_{\forall a^i} v_{a^i} \cdot \eta(e_{a^i})$$

The function $\eta(x)$ is defined as:

$$\eta(x) = \begin{cases} 2 & x > d \\ 1 & \frac{d}{2} < x \leq d \\ 0 & x \leq \frac{d}{2} \end{cases}$$

Where d is a constant parameter. We can extend the function $\eta(x)$ to add conditions for x.

6.3 Logarithm-Based Scoring Function

The logarithm-based scoring function focuses on the priorities (defined as v_{a^i}) more than on the basic scoring function. In particular, the precision of the top-priority attribute is of prime importance in the generalization. The logarithm-based scoring function is described as follows:

$$S_t = \max_{\forall a^i} v_{a^i} \cdot \frac{log(e_{a^i})}{log(e^b_{a_i})}$$

where $e^b_{a_i}$ is the value of e_{a_i} before anonymization. Figure 2 shows two tables generated by the η-based scoring function and the logarithm-based scoring function, respectively. The left table is generated using the logarithm-based scoring function with a priority that the attribute "Nationality" is the most valuable attribute in the table.

Table 1. Transaction Time of Pre-Sampling

Data	No. of Records	k'	Transaction Time
Adult	32,561	1	1289 ms
Adult	32,561	2	1299 ms
Adult	32,561	5	1307 ms
Adult	32,561	10	1286 ms
Census-income	199,523	1	4401 ms
Census-income	199,523	2	4324 ms
Census-income	199,523	5	4431 ms
Census-income	199,523	10	4464 ms

7 Performance Evaluation

We implemented a prototype system on a PC (Core i7 870 2.93 GHz, 4 GB Memory, Windows 7, 32 bit) and evaluated the transaction times for pre-sampling and anonymization. In this section, we show the results of the experiments.

We evaluated the prototype system using two data sets [5], *Adults Data Sets*, which has 32,561 records of 14 attributes, and *Census-Income Data Sets*, which has 199,523 records of 42 attributes, under several different sets of parameters k, l and c. These data sets had been used for performance evaluation in previous research. The transaction time for pre-sampling is shown in Table 1. The All threshold level of generalization for each attribute a^i was configured as 2 in the experiments. The All transaction time for *Adults Data Sets* was about 1.2 seconds and that for *Census-Income Data Sets* was about 4.4 seconds. The transaction times for anonymization for *Adults Data Sets* and *Census-Income Data Sets* are shown in Table 2 and Table 3, respectively. In all experimenst, we measured transaction time including the partial-domain de-generalization steps. The transaction time when using t-closeness was almost same as that when using l-diversity. There were no significant differences between the three scoring functions.

Thus, our prototype system is expected to generate anonymous tables in a feasible transaction time. In particular, where the table has fewer than 30,000 records and each consists of a reasonable number of attributes, the prototype system will generate an anonymous table for it in real-time.

8 Conclusion

In this paper, we proposed an anonymization scheme for generating a table with k-anonymity. The scheme calculates the scores of intermediate tables based on user-defined priorities for attributes, and from these it selects the table with the highest score. Three scoring functions were designed and they can be selected according to the specific requirements of each case. The generated table meets

Table 2. Transaction Time of Anonymization (Adult Data Sets)

Data	No. of Records	k	l	t	Scoring Function	Transaction Time
Adult	32,561	2	2	-	Basic	619 ms
Adult	32,561	2	2	-	*Log*-Based	592 ms
Adult	32,561	2	2	-	η-Based	680 ms
Adult	32,561	2	-	0.1	Basic	736 ms
Adult	32,561	2	-	0.1	*Log*-Based	720 ms
Adult	32,561	2	-	0.1	η-Based	675 ms
Adult	32,561	2	-	0.9	Basic	823 ms
Adult	32,561	2	-	0.9	*Log*-Based	778 ms
Adult	32,561	2	-	0.9	η-Based	723 ms
Adult	32,561	10	10	-	Basic	663 ms
Adult	32,561	10	10	-	*Log*-Based	631 ms
Adult	32,561	10	10	-	η-Based	669 ms
Adult	32,561	10	-	0.1	Basic	726 ms
Adult	32,561	10	-	0.1	*Log*-Based	722 ms
Adult	32,561	10	-	0.1	η-Based	696 ms
Adult	32,561	10	-	0.9	Basic	803 ms
Adult	32,561	10	-	0.9	*Log*-Based	777 ms
Adult	32,561	10	-	0.9	η-Based	833 ms

user's requirements and is employed in the services provided by users without any modification or evaluation. Our mechanism is applicable to full-domain and partial-domain generalization in some types of anonymity definitions to replace the check function. We will evaluate the prototype system using a number of tables for several users ' requirements, and consider an optimization method for parameters in our future work.

References

1. C. C. Aggarwal and P. S. Yu. On variable constraints in privacy preserving data mining. In *Proc. of the 5th SIAM International Conference on Data Mining*, pages 115–125, 2005.
2. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In *Proc. of ICDT 2005, LNCS*, volume 3363, pages 246–258, 2005.
3. G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Approximation algorithms for k-anonymity. In *Journal of Privacy Technology*, 2005.
4. S. S. Al-Fedaghi. Balanced k-anonymity. In *Proc. of WASET*, volume 6, pages 179–182, 2005.
5. A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.
6. R. J. Bayardo and R. Agrawal. Data privacy through optimal k-anonymity. In *Proc. of ICDE 2005*, pages 217–228, 2005.

Table 3. Transaction Time of Anonymization (Census-Income Data Sets)

Data	No. of Records	k	l	t	Scoring Function	Transaction Time
Census-Income	199,523	2	2	-	Basic	18585 ms
Census-Income	199,523	2	2	-	*Log*-Based	18381 ms
Census-Income	199,523	2	2	-	η-Based	16773 ms
Census-Income	199,523	2	-	0.1	Basic	19839 ms
Census-Income	199,523	2	-	0.1	*Log*-Based	19832 ms
Census-Income	199,523	2	-	0.1	η-Based	18525 ms
Census-Income	199,523	2	-	0.9	Basic	21889 ms
Census-Income	199,523	2	-	0.9	*Log*-Based	21363 ms
Census-Income	199,523	2	-	0.9	η-Based	19955 ms
Census-Income	199,523	10	10	-	Basic	22287 ms
Census-Income	199,523	10	10	-	*Log*-Based	21454 ms
Census-Income	199,523	10	10	-	η-Based	16319 ms
Census-Income	199,523	10	-	0.1	Basic	19788 ms
Census-Income	199,523	10	-	0.1	*Log*-Based	19871 ms
Census-Income	199,523	10	-	0.1	η-Based	18417 ms
Census-Income	199,523	10	-	0.9	Basic	25477 ms
Census-Income	199,523	10	-	0.9	*Log*-Based	24552 ms
Census-Income	199,523	10	-	0.9	η-Based	19040 ms

7. J.-W. Byun, A. Kamra, E. Bertino, and N. Li. Efficient k-anonymity using clustering technique. In *Proc. of the International Conference on Database Systems for Advanced Applications*, pages 188–200, 2007.
8. V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. k-anonymous data mining: A survey. In *Privacy-Preserving Data Mining: Models and Algorithms*. Springer-Verlag, 2008.
9. T. Dalenius. Finding a needle in a haystack —or identifying anonymous census record. In *Journal of Official Statistics*, volume 2(3), pages 329–336, 1986.
10. V. S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of ACM SIGKDD'02*, pages 279–288. ACM, 2002.
11. S. Kiyomoto, Y. Miyake, and T. Tanaka. Privacy Frost: A user-oriented data anonymization tool. In *Availability, Reliability and Security (ARES), 2011 Sixth International Conference on*, pages 442–447, 2011.
12. S. kiyomoto and T. Tanaka. A user-oriented anonymization mechanism for public data. In *Proc. DPM 2010, LNCS*, volume 6514, pages 22–35, 2010.
13. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proc. of SIGMOD 2005*, pages 49–60, 2005.
14. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *Proc. of the 22nd International Conference on Data Engineering (ICDE '06)*, pages 25–35. IEEE, 2006.
15. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Workload-aware anonymization. In *Proc. ACM SIGKDD'06*, pages 277–286. ACM, 2006.
16. J.-L. Lin and M.-C. Wei. An efficient clustering method for k-anonymization. In *Proc. of the 2008 international workshop on Privacy and anonymity in information society (PAIS '08)*, pages 46–50. ACM, 2008.

17. G. Loukides, A. Tziatzios, and J. Shao. Towards preference-constrained k-anonymisation. In *Database Systems for Advanced Applications: DASFAA 2009 International Workshops*, pages 231–245. Springer-Verlag, 2009.
18. A. Machanavajjhala, J. Gehrke, and D. Kifer. l-diversity: Privacy beyond k-anonymity. In *Proc. of ICDE'06*, pages 24–35, 2006.
19. A. Machanavajjhala, J. Gehrke, and D. Kifer. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Proc. of ICDE'07*, pages 106–115, 2007.
20. A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of PODS 2004*, pages 223–228, 2004.
21. J. Miller, A. Campan, and T. M. Truta. Constrained k-anonymity: Privacy with generalization boundaries. In *Proc. of the Practical Preserving Data Mining Workshop (P3DM2008)*, 2008.
22. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Trans. on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
23. P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information. In *Proc. of the 17th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (PODS'98)*, page 188, 1998.
24. X. Sun, M. Li, H. Wang, and A. Plank. An efficient hash-based algorithm for minimal k-anonymity. In *ACSC '08: Proceedings of the thirty-first Australasian conference on Computer science*, pages 101–107, 2008.
25. X. Sun, H. Wang, J. Li, T. M. Truta, and P. Li. (p^+, α)-sensitive k-anonymity: a new enhanced privacy protection model. In *Proc. of CIT'08*, pages 59–64, 2008.
26. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. In *J. Uncertainty, Fuzziness, and Knowledge-Base Systems*, volume 10(5), pages 571–588, 2002.
27. T. M. Truta and A. Campan. K-anonymization incremental maintenance and optimization techniques. In *Proceedings of the 2007 ACM symposium on Applied computing (SAC '07)*, pages 380–387. ACM, 2007.
28. T. M. Truta and B. Vinay. Privacy protection: p-sensitive k-anonymity property. In *Proc. of ICDE'06*, pages 94–103, 2006.
29. L. Willenborg and T. de Waal. *Elements of Statistical Disclosure Control*, volume 155. LNS, Springer-Verlag, 2001.
30. R. C.-W. Wong, J. Li, A. W.-C. Fu, and K. Wang. (α, k)-anonymity: an enhanced k-anonymity model for privacy preserving data publishing. In *Proc. of ACM SIGKDD'06*, pages 754–759, 2006.
31. X. Xiao and Y. Tao. Personalized privacy preservation. In *Proc. of SIGMOD 2006*, pages 229–240. ACM, 2006.
32. J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization for privacy preservation with less information loss. *SIGKDD Explor. Newsl.*, 8(2):21–30, 2006.
33. J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *Proc. of ACM SIGKDD'06*, pages 785–790. ACM, 2006.
34. H. Zhu and X. Ye. Achieving k-anonymity via a density-based clustering method. In *Proc. of APweb/WAIM 2007, LNCS*, pages 745–752. Springer-Verlag, 2007.

A Details of the Algorithm

Figure 3 shows details of the anonymization algorithm.

Input: a table T, k, l, c, t, p, H_{a^i}, $w^L_{a^i}$, v_{a^i}, $w^M_{a^i}$ (i=1, ..., n), *Score Function*
Output: T^G, s

*// *Pre-Sampling before Algorithm (Optional)*
//
// Precomputation:
Sort $(T, v_{a^1}, ..., v_{a^n})$
for $i = 1$ **to** n **do**
  $w_{a^i} \leftarrow w^L_{a^i}$
  while Check$(a^i) < k + p$ **do**
    $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    $w_{a^i} \leftarrow w_{a^i} - 1$
  end while
end for
while Check$_{k,l,c}(T) = NG$ and all $w_{a^i} > w^M_{a^i}$ **do**
  for $i = 1$ **to** n **do**
    if $w_{a^i} \geq w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i})
      $w_{a^i} \leftarrow w_{a^i} - 1$
    end if
  end for
end while
$T^I \leftarrow T$
$T^G \leftarrow T^I$
$s, s' \leftarrow$ Score(T)
if Check$_{k,l,c}(T) = NG$ **then**
  return *failed*
  end
else
  Top-Down Generalization
// Top-Down Generalization:
while *state* $\neq$ *stop* **do**
  $T' \leftarrow T^G$
  $s' \leftarrow s$
  state $\leftarrow$ *false*
  for $i = n$ **to** 1 **do**
    $T \leftarrow T'$
    $a^i \leftarrow$ De-Generalization(a^i, H_{a^i}, w_{a^i})
    if Check$_{k,l,c}(T) = OK$ and Score$(T) > s$ **then**
      $temp \leftarrow a^i, w_{a^i} + 1$
      $T^G \leftarrow T$
      $s \leftarrow$ Score(T)
      state $\leftarrow$ *true*
    end if
  end for
  if *state* $=$ *false* **then**
    state $\leftarrow$ *stop*
  end if
  $a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow temp$
end while
return T^G, s **repeat**
// 2nd Top-Down:
$T' \leftarrow T^G$
$s' \leftarrow s$
state $\leftarrow$ *false*
for $i = n$ **to** 1 **do**
  $T \leftarrow T'$
  $a^i \leftarrow$ De-Generalization(a^i, H_{a^i}, w_{a^i})
  if Score$(T) > s'$ **then**
    $temp \leftarrow a^i, w_{a^i} + 1$
    $T^T \leftarrow T$
    $s^T \leftarrow$ Score(T)
    state $\leftarrow$ *true*
  end if
end for
if *state* $=$ *false* **then**
  state $\leftarrow$ *stop*
  return T^G, s
else
// Bottom-Up:
while Check$_{k,l,c}(T) = NG$ **do**
$T' \leftarrow T^T$
state $\leftarrow$ *false*
$a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow temp$
  for $i = 1$ **to** n **do**
    $T \leftarrow T'$
    if $w_{a^i} > w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    end if
    if Check$_{k,l,c}(T) = OK$ and Score$(T) > s$ **then**
      $temp \leftarrow a^i, w_{a^i} - 1$
      $T^G \leftarrow T$
      $s \leftarrow$ Score(T)
      state $\leftarrow$ *true*
    end if
  end for
  if *state* $=$ *true* **then**
    $a^1, ..., a^n, v_{a^1}, ..., v_{a^n} \leftarrow temp$
  else
  for $i = 1$ **to** n **do**
    $T \leftarrow T'$
    if $w_{a^i} > w^M_{a^i}$ **then**
      $a^i \leftarrow$ Generalization(a^i, H_{a^i}, w_{a^i})
    end if
    if Score$(T) > s'$ **then**
      $T^T \leftarrow T$
      $temp \leftarrow a^i, w_{a^i} - 1$
      $s' \leftarrow$ Score(T)
      state $\leftarrow$ *true*
    end if
  end for
  if *state* $=$ *false* **then**
    state $\leftarrow$ *stop*
  end if
end while
until *state* $=$ *stop*
// Partial-Domain Generalization:
while Check$_{k,l,c}(T^P) = NG$ **do**
  $T^G \leftarrow T^P$
  $T^P =$ Pertial-Domain De-Generalization(T^G)
end while
return T^G, s

Fig. 3. Algorithm