ANON - a flexible tool for achieving optimal k-anonymous and ℓ-diverse tables *

Margareta Ciglic, Johann Eder, and Christian Koncilia

Alpen-Adria-Universität Klagenfurt,
Department of Informatics Systems,
Klagenfurt, Austria
{firstname.lastname}@aau.at

Abstract. Anonymization of data is an indispensable task when sensible data has to be shared between different organizations, e.g. when sharing genomic data gathered at and shared between different medical research organizations. Several concepts have been discussed during the last couple years, for instance k-anyonymity and ℓ-diversity. Both concepts are privacy constraints that must be guaranteed in query results to assure a high privacy degree. However, the approaches presented so far cannot deal with NULL values in the underlying data sources. In this technical report we will discuss the problem of NULL values for anonymization in detail and present and discuss our approach for k-anonymity which explicitly considers NULL values. Furthermore, we will present our implementation of this approach.

Keywords: privacy, microdata, k-anonymity, ℓ-diversity, NULL values, missing values

1 Introduction

Anonymization of data is an indispensable task when sensible data or microdata (any collection of data with detailed information on entities) has to be shared between different organizations, e.g. when sharing genomic data gathered at and shared between different medical research organizations. Several concepts have been discussed during the couple last years, for instance k-anyonymity and ℓ-diversity (an overview over these concepts is given in [10]).

Furthermore, for data collections requiring the willingness of data owners to share (donate) their data, studies [6] clearly indicate that the protection of privacy is one of the major concerns of data owners and decisive for a consent to donate data [12]. For protecting privacy from linkage attacks the concept of k-anonymity received probably the widest attention. Its core idea is to preserve privacy by hiding each individual in a crowd of at least k members. Many anonymization algorithms implementing these concepts were developed.

*The work reported here was supported by the *Austrian Ministry of Science and Research* within the project BBMRI.AT and the *Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V. (TMF)* within the project ANON.

Surprisingly, neither the original definition of k-anonymity nor any of the many anonymization algorithms deals with unknown, or missing values (NULL values in database terms) in microdata. We could not find a single source discussing the problem of NULL values in microdata for anonymization. Recent surveys [21] or textbooks [11] do not mention NULL values or missing values. However, all techniques and algorithms we found, explicitly or implicitly require that all records with at least one NULL value have to be removed from a table before it can be anonymized ([18,15,14,28,25,31,13,3], and many more). There is only some treatment of NULL values in form of suppressed values, i.e. NULL values resulting from removing ("suppressing") data in the course of anonymization procedures. Attack possibilities on suppressed rows can be found in [29] and [24]. A discussion about suppression of values in single cells can be found in [5,22,1]. However, neither of these approaches discusses the problem of missing values in the original data or of non-existing values due to non-applicable attributes.

NULL values, nevertheless, are not exceptional in microdata, e.g. they appear frequently in datasets for medical research [8,7,32]: Some attributes might not be applicable for each patient. A patient might have refused to answer some questions in a questionnaire or could not be asked due to physical or mental conditions. In an emergency situation some test might not have been performed, etc. etc.

Anonymization by generalization and suppression of data cause loss of information. The aim of reducing this information loss triggered many research efforts. The ignorance of NULL values in anonymization algorithms results in dropping rows from a table, causing a considerable loss of information. Furthermore, dropping rows with NULL values also could introduce some bias in the dataset which is not contained in the original table. This is of course unfortunate for further analysis of the data (for example in evidence based medicine) and might compromise the statistical validity of the results. For example, dropping rows with a NULL value in the field *occupation* would skip all children from the data set and introduce an age bias which was not present in the original dataset.

In [4], we provided a thorough grounding for the treatment of NULL values in anonymization algorithms. Furthermore, we showed that we can reduce the problem of NULL values in k-anonymity to different definitions of matching between values and NULL values. In this technical report we will discuss the implementation of our approach, named ANON, in detail.

This technical report is organized as follows: in section 2 we will give an overview of k-anyonymity and ℓ-diversity. Furthermore the concepts presented in [4] will be discussed. In section 3 we will discuss the ANON Anonymization algorithms in detail. Section 4 will discuss the ANON implementation. Evaluation results and experiments will be presented in section 5. Finally, conclusions will be given in section 6.

2 Basic Terms and Techniques

2.1 k-anonymity and ℓ-diversity

Data privacy is often one of the most important requirements when dealing with data. This is especially the case if personal data is the subject matter. For this purpose several privacy preserving techniques have been introduced in the literature. Matthews and Harel describe in [20] the major methods for assessing privacy. In this work we focus on the anonymization approach, in particular the combination of k-anonymity and ℓ-diversity. An overview over the privacy models of anonymization approach is given in [10].

k-Anonymity and ℓ-diversity are two privacy constraints that must be guaranteed in microdata to assure a high privacy degree. k-Anonymity[26] is checked on quasi identifiers and ℓ-diversity[19] on sensitive attributes, after unique identifiers (e.g. social security number, full name, etc.) have been removed from the data. Table 1 shows an example of such microdata and groups the attributes into (1) unique identifiers, (2) quasi-identifiers or non-sensitive attributes and (3) sensitive attributes.

Unique identifier	**Quasi-identifiers**			**Sensitive attribute**
Name	ZIP	Age	Sex	Condition
Alice	13053	28	F	Hepatitis
Bob	13068	29	M	Hepatitis
Charlie	13068	21	M	Flu
Daniel	13053	23	M	Flu
Emma	14853	50	F	Cancer
Felix	14853	55	M	Hepatitis
George	14850	47	M	Flu
Harry	14850	49	M	Flu
Isabel	13053	31	F	Cancer
Jenny	13053	37	F	Cancer
Kate	13068	36	F	Cancer
Lara	13068	35	F	Cancer

Table 1. Example of microdata in medical domain (adapted from [19]).

Unique identifiers, as indicated by the word itself, are a group of identifiers that uniquely identify a person. Social security numbers and other similar identifiers belong to this group. Quasi-identifiers (e.g. age, height) per definition do not identify a person, but a set of them together can become an identifier when they are linked to some other data source that contains the same quasi identifiers and additionally unique identifiers. Such linking of 2 sources is called a linking attack [29]. The linking attack can be successfully prevented with k-anonymity of microdata. k-Anonymity assures that each individual (record) is hidden in a

crowd of k individuals with identical quasi identifier values. In a table with microdata, we call a crowd of people that do not distinguish in their quasi identifiers a partition, a group or a block.

k-Anonymity only checks the quasi identifiers of individuals, but not the sensitive part of the data – the sensitive attributes. Those attributes reveal very sensitive information about individuals, e.g. data about donors health or lifestyle. This can be a diagnosis, sexual orientation, drug consumed, etc. Exactly this information is essential for research, but must not be linked to an individual as required by legal restrictions, contracts or ethical bylaws. The rights and dignity of donors must be guaranteed. This guarantee can be assured with the combination of k-anonymity and ℓ-diversity. ℓ-Diversity proves the sensitive attributes and checks if a partition with k records also contains ℓ different values of the sensitive attributes. This constraint prevents attacks on sensitive attributes like the homogeneity attack. Homogeneity attack[19] happens if all records within a partition have the same value of a sensitive attribute. For example all patients from a partition with k or more records suffer from cancer. If we know that somebody is "hidden" in this partition, we also know that he suffers from cancer, because everybody in this partition suffers from cancer.

Table 2 below shows 2 anonymized versions of the initial table 1. The left table is 4-anonymous, but it does not satisfy any reasonable diversity. Therefore the last partition with the individuals $\{I, J, K, L\}$ is not secure in respect to homogeinity attack. The right table is 4-anonymous and additionally 3-diverse. No described attacks can occur in this anonymized table version.

	ZIP	Age	Sex	Condition
A	130**	< 30	*	Hepatitis
B	130**	< 30	*	Hepatitis
C	130**	< 30	*	Flu
D	130**	< 30	*	Flu
E	148**	≥ 40	*	Cancer
F	148**	≥ 40	*	Hepatitis
G	148**	≥ 40	*	Flu
H	148**	≥ 40	*	Flu
I	130**	3*	*	Cancer
J	130**	3*	*	Cancer
K	130**	3*	*	Cancer
L	130**	3*	*	Cancer

	ZIP	Age	Sex	Condition
A	1305*	≤ 40	*	Hepatitis
D	1305*	≤ 40	*	Flu
I	1305*	≤ 40	*	Cancer
J	1305*	≤ 40	*	Cancer
E	1485*	> 40	*	Cancer
F	1485*	> 40	*	Hepatitis
G	1485*	> 40	*	Flu
H	1485*	> 40	*	Flu
B	1306*	≤ 40	*	Hepatitis
C	1306*	≤ 40	*	Flu
K	1306*	≤ 40	*	Cancer
L	1306*	≤ 40	*	Cancer

Table 2. 4-Anonymous table (left) and a 4-anonymous and 3-diverse table (right) (adapted from [19]).

2.2 Generalization and Suppression

Above, we described the basic terms and 2 major anonymization constraints k-anonymity and ℓ-diversity. Now, let us describe 2 basic techniques, *generalization* and *suppression*, that are commonly used to achieve k-anonymity and ℓ-diversity. Both techniques decrease information content of the data to meet the required privacy degree. They can be used isolated or in combination.

Generalization[26,29] replaces the values of quasi identifiers (QID) with more general values. In order to do this, one must build generalization hierarchies (taxonomy trees or intervals with step definitions) for all QIDs. Generalization hierarchies define a transformation for each possible value from the domain with the most detailed (original) values over the more general domains to the top domain with a single, most general value. The generalization hierarchy and its corresponding domains of the QID *ZIP* of the running example (tables 1 and 2 are shown in figure1.

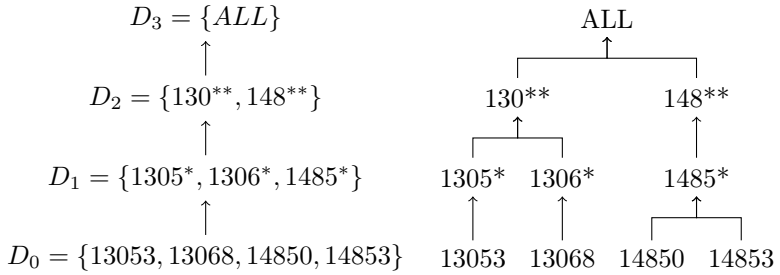

Fig. 1. Generalization hierarchy and its corresponding domains (generalization levels) of the QID *ZIP* of the running example

When we use generalization, we can use 2 types of recoding (value transformation): *global recoding* or *local recoding*. If all values of a QID are transformed to the same generalization level, we speak of *global recoding* or *full-domain generalization*. In local recoding, the values of a QID can be transformed to different generalization levels (in some partitions to a higher level, in some to a lower). Even one value that occurs several times in the microdata can be transformed to different levels in different partitions. Both recoding methods are described in detail in [30].

In opposition to generalization, suppression does not transform the values to other, more general values, but simply deletes (eliminates) them. Suppression can be undertaken on single values (called cell suppression), on whole tuples (called tuple suppression) or on whole attributes (called attribute suppression). The impact of the attribute suppression is the same as the one of the generalization of an attribute to the top level, where the data loss is 100%. Cell suppression

is rarely used for achieving k-anonymity due to its complexity. Approximation algorithms that use cell suppression are described in [23] and [2]. Tuple suppression can not be used alone for achieving k-anonymity, but it was found as very useful together with generalization. In combination with generalization[26,29], tuple suppression can be used to eliminate the outlier tuples, before the remaining tuples get generalized. Outlier tuples are those which hardly match any other tuples and therefore push the generalization levels even up to the top level to meet the required k-anonymity and ℓ-diversity. A single green eyed person in a microdata with only blue or brown eyed people is an outlier.

ANON uses full-domain generalization, paired with tuple suppression.

3 ANON Anonymization Algorithm

ANON is based on Priority-based anonymization algorithm[27] that allows users to specify priorities for the generalization of quasi identifiers and to set the generalization limits. Similar to the Priority-based algorithm, ANON uses the priorities of QIDs to calculate the weighted information loss of an anonymized table. This weighted information loss is used in ANON's best-first search to determine the best anonymized table. ANON can check k-anonymity as well as ℓ-diversity. The main contribution of ANON is handling of missing values.

3.1 Missing Value Handling

We discussed NULL values in detail in [4] and introduced 4 different matches for QIDs: *basic match*, *extended match*, *maybe match* and *right maybe match*. Basic match does not handle NULL values at all, but eliminates them instead. In basic match definition, two tuples match if they contain identical QID values - but not NULL values. Extended match handles NULL values as normal values (two NULL values match each other). Maybe match and right maybe match go a step further and notice NULL values as wildcards that match every value. However, maybe match and right maybe match are not safe against attacks on NULL values that we introduced in [4].

ANON's goal is to guarantee the highest possible privacy degree and at the same time the best possible data quality. Therefore it implements beside the standard basic match also the extended match. The user can decide whether NULL values should be handled or not. The algorithm 1 below shows how basic match or extended match are used to build partitions that are used afterwards for the privacy check.

Used variables have following meaning:

- $null\_handling$: denotes if NULL values are handled or not. If $null\_handling$ is TRUE than the partitioning algorithm will use extended match, otherwise the basic match.
- $null\_string$: represents the NULL value string - mostly $NULL$.
- *partition*: a partition is a set of tuples that match each other.

- *partitionset*: set of partitions.
- *table*: a table with microdata.
- *tuple*: a table row.
- *matched*: denotes if a tuple matches a partition.

Algorithm 1 ANON's Partitioning Algorithm

Input: *table*, *null_handling*, *null_string*

Output: set of partitions *partitionset*, where each partition contains tuples with identical values of quasi identifiers or NULLs instead of a quasi identifier value. Partitions are disjoint.

```
1: function PARTITION-TABLE(table, null_handling, null_string)
2:     partitionset ← { }
3:     for each tuple in table do
4:         matched ← FALSE
5:         for each partition in partitionset do
6:             if tuple matches partition then        ▷ tuple has identical quasi identifiers
                                                      ▷ as other tuples in partition or a quasi identifier
                                                      ▷ is NULL in the tuple and all other tuples in partition
7:                 add tuple into partition
8:                 matched ← TRUE
9:             end if
10:        end for
11:        if matched = FALSE then
12:            if tuple does not contain NULL or null_handling = TRUE then
13:                partition ← {tuple}
14:                add partition into partitionset
15:            end if
16:        end if
17:    end for
18:    return partitionset
19: end function
```

After a table has been partitioned, ANON executes the privacy test on each partition. Until a secure solution is found, ANON searches it among all possible generalized tables.

3.2 Search Algorithm and Privacy Test

ANON's anonymization algorithm uses best-first search algorithm to find the optimal solution and weighted information loss to evaluate the cost of a potential solution (generalized table). Weighted information loss is calculated with the formula $\text{WIL} = \sum_{i=1}^{n} prios[i] * \varphi_{\alpha_i}^{levels[i]}$, where n is the number of quasi

identifiers, $\varphi_{\alpha_i}^{levels[i]}$ the loss of information if the attribute α_i gets generalized to the level $levels[i]$ and $prios[i]$ is the priority of the attribute α_i.

The algorithm consists of 2 main parts: table search (function ANONYMIZE-TABLE) and privacy test (function PRIVACY-TEST).

The main function is ANONYMIZE-TABLE which takes the following parameters as input:

- *table*: original table that has to be anonymized,
- *limits*: array with generalization level limits for all quasi identifiers,
- *prios*: array with priorities for all quasi identifiers,
- *k_param*: k - the minimal partition size,
- *l_params*: array with minimal required diversities for all sensitive attributes,
- *max_supp*: number of tuples that are allowed to be suppressed.

Variables and values used in algorithms 2 and 0:

- *open*: List of potential solutions (generalized tables).
- *visited*: List of potential solutions that have already been added to *open*.
- *best*: Generalized table from *open* with the lowest information loss.
- *levels*: Generalization levels of a potential solution.
- *child*: Child node - a potential solution with a table generalized to the next higher level at one quasi identifier while the other attributes' levels remain the same as the levels of the parent.
- *supp_tuples*: Number tuples that violate the privacy constraints.
- *partition*: Table partition - a set of records with the same quasi identifier values.
- *diversities*: Array with diversities of one partition for all sensitive attributes.
- NIL: represents a NULL value.
- FAILURE: denotes that an anonymized table which satisfies all constraints could not be found.

Instance variables of a node used in algorithms 2 and 0:

- *node*.PARENT: Parent node - potential solution from which *node* was deduced.
- *node*.LEVELS: Generalization levels of *node*.TABLE. Represents action.
- *node*.WIL: Weighted information loss. Represents total path cost.
- *node*.TABLE: Table with values generalized to *node*.LEVELS. Represents state.

Functions used in algorithms 2 and 0:

ANONYMIZE-TABLE Input: *table*, *limits*, *prios*, *k_param*, *max_supp*
Output: Anonymized table which satisfies all input constraints or FAILURE if no solution could be found.

GROUP-TABLE Input: *table*
Output: set of groups *groupset*, where each group contains tuples with identical values of quasi identifiers or NULL instead of a quasi identifier value. Groups may not be disjoint, since tuples that contain NULL values may occur in more than one group.

Privacy-Test Input: *node*, *k_param*, *max_supp*
Output: TRUE if *node*.Table satisfies all privacy constraints, else FALSE.
Make-Node Input: *table*
Output: Node that represents *table* in the search space.
Generalize Input: *table*, *node*.Levels
Output: Table with values generalized to node.Levels.
Child-Node Input: *parent*, *levels*, *prios*
Output: Node with Parent = *parent*, Levels = *levels*.
Calculate-WIL Input: *levels*, *prios*
Output: Weighted information loss of a generalized table, calculated as follows:
$\sum_{i=0}^{length(levels)-1} prios[i] * \varphi_{\alpha_i}^{levels[i]}$ $\varphi_{\alpha_i}^{levels[i]}$ is the loss of information if the attribute α_i gets generalized to the level $levels[i]$.
Count-Tuples Input: *partition*
Output: Number of tuples that there are in the *partition*.
length Input: *array*; Output: Length of the *array*.

ANON is meant to be a customizable tool – for the user as well as for the developer – therefore the implementation offers an abstract class of search algorithms that can easily be extended by new search algorithms. The privacy check does not have to be written again, because it is decoupled from the search algorithm. Similar interfaces are provided for information loss calculation, as well as for ℓ-diversity check. Details about the implementation are given in the next section.

Algorithm 2 ANON's Priority-based Algorithm - Part 1 (Search Algorithm)

Input: *table*, *limits*, *prios*, *k_param*, *l_params*, *max_supp*
Output: anonymized table satisfying k-anonymity and ℓ-diversity or FAILURE if no solution could be found

```
1: function ANONYMIZE-TABLE(table, limits, prios, k_param, l_params, max_supp)
2:     open ← {MAKE-NODE(table)}
3:     visited ← { }
4:     while open is not empty do
5:         best ← node n in open with the lowest n.WIL value
6:         if best.TABLE = NIL then
7:             best.TABLE ← GENERALIZE(table, best.LEVELS)
8:         end if
9:         if PRIVACY-TEST(best, k_param, l_params, max_supp) then
10:            return best.TABLE
11:        else                                              ▷ expand best
12:            for i ← 0 to length(limits) − 1 do
13:                levels ← best.LEVELS
14:                if limits[i] > levels[i] then
15:                    levels[i] ← levels[i] + 1
16:                    child ← CHILD-NODE(best, levels, prios)
17:                    if child not in visited then
18:                        add child into open
19:                        add child into visited
20:                    end if
21:                end if
22:            end for
23:            best.TABLE ← NIL
24:            remove best from open
25:        end if
26:    end while
27:    return FAILURE
28: end function

29: function CHILD-NODE(parent, levels, prios)
30:    return a node with
31:            PARENT ← parent,
32:            LEVELS ← levels,
33:            WIL ← CALCULATE-WIL(levels, prios)          ▷ Weighted Information Loss
34:            TABLE ← NIL
35: end function
```

Algorithm 3 ANON's Priority-based Algorithm - Part 2 (Privacy Test)

```
36: function PRIVACY-TEST(node, k_param, l_params, max_supp)
37:     supp_tuples ← 0
38:     for each partition in PARTITION-TABLE(node.TABLE) do
39:         if COUNT-TUPLES(partition) ≥ k_param then        ▷ k-anonymity satisfied
40:             diversities ← CALCULATE-DIVERSITIES(partition)
41:             for i ← 0 to length(l_params) − 1 do
42:                 if diversities[i] < l_params[i] then        ▷ ℓ-diversity not satisfied
43:                     remove partition from node.TABLE
44:                     supp_tuples ← supp_tuples + COUNT-TUPLES(partition)
45:                     break
46:                 end if
47:             end for
48:         else                                        ▷ k-anonymity not satisfied
49:             remove partition from node.TABLE
50:             supp_tuples ← supp_tuples + COUNT-TUPLES(partition)
51:         end if
52:         if supp_tuples > max_supp then              ▷ privacy not satisfied
53:             return FALSE
54:         end if
55:     end for
56:     return TRUE          ▷ privacy satisfied (supp_tuples ≤ max_supp)
57: end function
```

4 ANON Principle and Implementation

ANON is a flexible and customizable anonymization tool implemented in Java. It is available in two distributions: as a Web Service and as an executable platform-independent java archive (JAR) with a simple graphical user interface.

ANON's guiding principal is to offer an anonymization software solution which produces best quality results, is highly customizable and can be used in a wide application area. Its application domain can be switched immediately without any changes in ANON. This possibility is enabled by the ANON definition file, which is a combination of an anonymization settings file and a metadata file. As drawn in figure 2, ANON definition file is the main ANON input that determines the microdata source(s) and the outputs, as well as the anonymization process itself.

ANON definition is an XML file that consists of the following five fundamental sections:

Parameters define the anonymization settings (the value of k, maximal suppression threshold max_supp, type of the search algorithm, ANON report settings and missing value handling details).

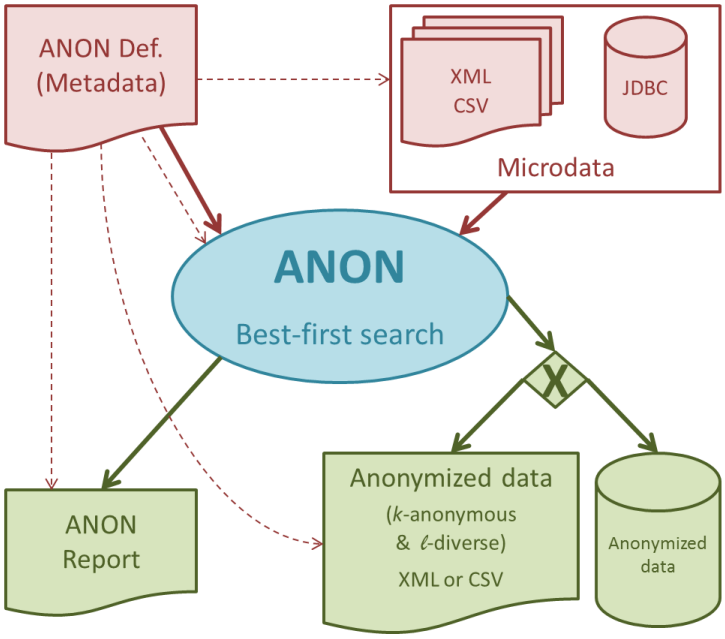

Fig. 2. ANON principle

Datasource definition defines the source(s) of data that should be anonymized (database(s), XML or CSV-file(s)).

Output definition defines the target where the anonymized data should be saved (database, XML or CSV-file).

Attributes definition defines which attributes should be read from the source data and how these should be handled. Each k-attribute should have assigned its priority and maximal generalization limit. Each l-attribute must be configured with the ℓ-diversity type that should be used for checking, and its desired parameter values (ℓ).

Generalization hierarchies define value generalization hierarchies that are used for anonymization. For every quasi identifier attribute (k-attribute) there should be one generalization hierarchy, upon which the values are generalized. Each generalization hierarchy contains information about the hierarchy levels inclusive their information loss and a value generalization tree. This tree must contain all the values that the corresponding quasi identifier can hold in the microdata.

ANON definition file is maintained by the user and can by adapted to his/her needs anytime. The XML Schema file ANONSchema.xsd defines a valid ANON definition file.

As noticeable from the listings above, ANON is capable of anonymizing data from multiple sources that have one of the following formats: database connection

(JDBC), XML-file or CSV-file. The result can be saved as one of these formats as well. Besides the anonymization outcome, user can decide to receive an ANON report, which informs about the anonymization process and eventual failures.

ANON offers the user the possibility to select the attributes that the user wants to handle in the anonymization process and mark them with one of the following anonymization types:

- k-attribute,
- l-attribute,
- dontcare,
- ignore.

The attributes that should be skipped from the result should be marked with "ignore". Alternatively they can be left out of the ANON definition file to raise the same effect. If an attribute does not play any role for the individuals privacy and should appear unchanged in the result, then it should be marked as "dontcare". The remaining two types are those that are relevant for the anonymization process.

Attributes marked with "k-attribute" are quasi identifiers that must be transformed to a particular generalization level, such that k-anonymity for the whole table is satisfied. For this kind of attributes, the user should also specify the generalization limit and attribute priority. For l-attributes (sensitive attributes), the ℓ-diversity type and its parameter(s) should be defined.

ANON is designed to provide anonymized tables with multi-attribute ℓ-diversity. The ℓ-diversity can be defined on the attribute level, not globally as the parameter k is. Furthermore, ANON allows to assign different ℓ-diversity types to different attributes. This is a noteworthy advantage of ANON, because none of ℓ-diversity types can be appropriate for all kinds of sensitive attributes' values. For example, distinct ℓ-diversity with $\ell = 2$ performs well when checking an attribute with only two possible values that are equally distributed (e.g. IgG test for a common infection with possible values *positive* and *negative*). The situation is not so optimistic if the same ℓ-diversity must prove the diversity of some numerical values (e.g. income). Such values are presumably highly diverse, but may semantically not differ much. It is a big difference between two incomes €1,000 and €4,000 and barely a difference between €1,000 and €1,050, but both pairs are diverse. For values of this kind, a distance measure would provide much higher privacy.

ANON responded to this observation with an ℓ-diversity interface, which allows to add new ℓ-diversity types and their calculations without changing ANON's source code. Such interfaces are also available for the search algorithm, input/output data format, etc. At the moment, ANON only supports distinct ℓ-diversity, but has already prepared instantiations for entropy ℓ-diversity and recursive ℓ-diversity, where the diversity calculation must be implemented.

Instead of ℓ-diversity, t-closeness[17] can be integrated into ANON without a need to change the code, since ANON's ℓ-diversity interface requires to pass a list of all values of a sensitive attribute that appear in the group to the ℓ-diversity

calculation method. To add t-closeness to ANON, an additional ℓ-diversity class to calculate the metric could be written with ease.

5 ANON Evaluation and Experiments

ANON's anonymization algorithm (a best-first search instantiation) is optimal and complete. If generalization limits are set to less than the number of generalization levels, it is possible that the algorithm will not find a solution (because there is none). If no limits are set, it always finds a solution, which is in worst case a completely generalized table with only one partition.

ANON is NP-complete. Its worst case time complexity equals to the number of nodes in the state space, which grows exponentially with the number of quasi identifier attributes. State space size is calculated with the formula $\Pi_{i=1}^{n} limits[i]$, where n is the number of quasi identifiers and $limits[i]$ is the generalization limit of the quasi identifier α_i.

Space complexity is almost constant because the solutions' states (generalized tables) are not kept in the memory. After a state has been evaluated by the function Privacy-Test, it is immediately removed from the memory. The original table is kept in the memory until the end of an anonymization process, so there are at most two tables kept in the memory at the same time.

Although ANON is NP-complete and its time complexity in $\mathcal{O}(2^n)$, where n is the number of quasi identifiers, it is a feasible (practicable) solution, because the number of quasi identifiers in a table is low and limited and the number of tuples nearly does not have any impact on the time complexity. Furthermore, the users do not require any real time solutions. Tables are not anonymized very frequently, so it is tolerable, if it takes a few minutes to generate a high quality solution. For the researchers, the utility of data is more important though.

Detailed experiments on complexity and other important characteristics of ANON have shown that ANON's performance is good anyhow and can additionally be influenced by many parameters, which can be modified by the user. The experiments are presented in the next section.

5.1 Experiment Data and Setup

All experiments described in coming sections were performed on a computer with an Intel® Core™2 Quad CPU Q9400 @ 2.66GHz processor, 8GB RAM and Windows 7 Enterprise operating system.

The microdata used for the experiments was the *Adult Data Set* from UCI Machine Learning Repository[9], which is commonly used for performance experiments in the microdata privacy literature. Adult Data Set contains real data collected by the U.S. census bureau in the year 1994. This data is split in a training set and a test set. For our experiments we merged both sets together and tuples with unknown values were removed. After data cleaning, the set contained 45,222 tuples. To provide comparable results, same data preparation was undertaken as described in [19] and [16]. From the 15 attributes in the data set,

the identical nine were chosen as in [19]. As shown in table 3, the attributes *age*, *gender*, *race*, *marital status*, *education*, *native country* and *workclass* were used as quasi identifiers and the attributes *salary class* and *occupation* were used as sensitive attributes. Generalization hierarchies for the used quasi identifiers were constructed in a semantically logical way. They come with the ANON distribution.

	Attribute	**Domain size**	**Generalization type**	**No. of gen. levels**
1	Age	74	Ranges (5, 10, 20, 100)	4
2	Gender	2	Taxonomy tree	1
3	Race	5	Taxonomy tree	1
4	Marital Status	7	Taxonomy tree	2
5	Education	16	Taxonomy tree	3
6	Native Country	41	Taxonomy tree	2
7	Work Class	7	Taxonomy tree	2
8	Salary class	2	Sensitive att.	
9	Occupation	14	Sensitive att.	

Table 3. Adult Data Set description (adapted from [19])

There were two experiment runs, each with almost 800 anonymizations: one with the use of priorities and information loss and one without them, to imitate the optimal search algorithms without a cost function (e.g. MinGen). Information loss values are listed within generalization hierarchies that come with ANON. The priority order of attributes (starting with a low priority) in the first run was {*age*, *native country*, *education*, *marital status*, *workclass*, *race*, *sex*}. In the second run (without the cost function), an implicit priority order was derived from the attributes order in the ANON definition file, which was {*age*, *sex*, *race*, *marital status*, *education*, *native country*, *workclass*}. Generalization limits were not set (they equaled to the no. of generalization levels) to avoid an anonymization without a solution.

The experiments were performed to estimate the "real case" complexity and the impact of different parameters on the number of visited nodes, resulting information loss and average partition size. These parameters are listed in table 4. There were over 1,500 produced anonymizations, where the parameters were set to almost all possible value combinations.

As parameter values, the representative values according to the results of pre-experiments were chosen. Those experiments have shown that, for example, the skipped k values (4, 6, 7, 8, 9, 11,...) do not provide different performance results as the next higher (or lower) chosen value. Some of the representative parameter value combinations and their results are drawn in the charts in the following sections. Each line point (marker) in a chart represents one anonymization setting (or simply one anonymization). The line marker style is mostly chosen pairwise,

Parameter	Chart notation	Values
n	QID	1, 2, 3, 4, 5, 6, 7
k	k	2, 3, 5, 10, 14, 20, 100, 200, 1000
ℓ_{α_8}	l1	1, 2
ℓ_{α_9}	l2	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
max_supp	supp.	0%, 1%, 10%
cost function (WIL)	with priorities	used (first run), not used (second run)
	w/o priorities	

Table 4. Experiments' parameters and their values

such that a line with black markers has something in common with a line with the same, but white, markers. The same applies for marker shape. The dotted red lines always represent some upper or lower bound. Charts with the parameter k on the x-axis that contain a setting "2 sensitive", assign $\ell_{\alpha_8} = 2$ (l1=2) and adapt parameter ℓ_{α_9} to parameter k (l2=k) until the limit $\ell_{\alpha_9} = 14$ is reached.

5.2 Algorithm Complexity and Impact of Quasi Identifiers' Size

The first two experiments deal with the time complexity. In these experiments, the impact of the quasi identifiers' increase on the number of visited nodes and anonymization time was analyzed. Both experiments were executed with four different anonymization settings groups: 1) k=2 with 0% tuple suppression (blue line), 2) k=2 with 1% suppression limit (violet line), 3) k=10 with 0% suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 10$ (green line) and 4) k=10 with 1% suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 10$ (orange line). The red dotted line denotes the maximal possible number of visited nodes ($\Pi_{i=1}^{n} limits[i]$) and the approximated maximal required anonymization time, respectively. As approximation, the anonymization settings with k=14, 0% suppression and $\ell_{\alpha_8} = 2$, $\ell_{\alpha_9} = 14$ were used. These settings were noticed to result in maximal possible values, because only the last queued node with completely generalized table satisfies these settings.

It is commonly known that time complexity grows with the number of quasi identifiers. In contrast to that, the number of tuples in a table does not have a big impact on time complexity. Table size is just a constant factor multiplied by the number of nodes, which does not affect the number of visited nodes itself and can therefore be neglected. Figure 3 confirms for ANON that time complexity grows with the increase in quasi identifiers.

Figure 4 represents the same experiment, where, instead of number of visited nodes, the time was measured. If we compare both figures, it is easy to see that time depends on the number of nodes and some other minor factors. If we observe the ℓ-diversity lines (green and orange lines) in figures 3 and 4, we can notice that these lines have a slightly higher slope in the chart with time on the y-axis than in the chart with nodes on the y-axis. The explanation for this phenomenon is hidden in diversity calculation effort. ANON does not need to calculate the size of a partition (relevant for k-anonymity checking) explicitly,

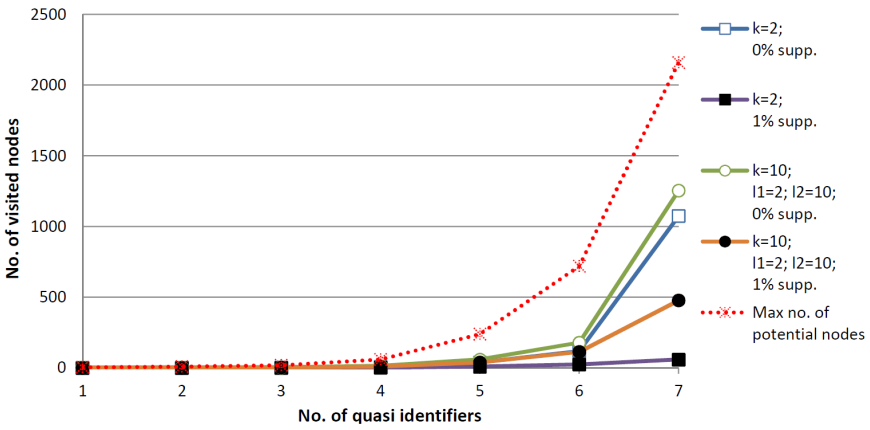

Fig. 3. Search algorithm complexity (number of nodes)

because it is managed together with a partition. The diversity (relevant for ℓ-diversity checking), in opposition to partition size, has to be calculated extra for each partition, if the partition is k-anonymous.

A very interesting and important piece of information visible in charts 3 and 4, is the enormous impact of tuple suppression. The blue and the violet line have the same settings, except the maximal suppression limit (blue 0%, violet 1%). However, the difference in the results is huge. It took 1,075 nodes to anonymize 7 quasi identifiers to a 2-anonymous table with no suppression (blue) and with just 1% suppression (max. 453 tuples may be eliminated), it took only 60 nodes until the optimal solution was found.

5.3 Impact of Parameter Values

Parameter k is the central parameter in microdata anonymization. It determines the minimal partition size, but (of course) also the maximal ℓ parameter. Figure 5 shows how the number of visited nodes increases with a rising k. The lines with black markers represent anonymizations that had to satisfy only k-anonymity. The lines with white markers had to satisfy ℓ-diversity, too ($\ell_{\alpha_8} = 2$; $\ell_{\alpha_9} = k$ or $\ell_{\alpha_9} = 14$ if $k > 14$).

The chart shows that the number of nodes does not differ very much between anonymization with only k-anonymity constraint and those with added ℓ-diversity – until $k = \ell = 10$. At this point, the ℓ-lines jump dramatically. This jump is explained with the number of distinct values of the second sensitive attribute *occupation* – which is 14. If those 14 values would be equally distributed, the problem would not be so hard, but they are not. A detailed data analysis showed that one value of attribute *occupation* (Armed-Forces) appears only 14-times. That means that maximal 14 partitions are possible at all. Since solu-

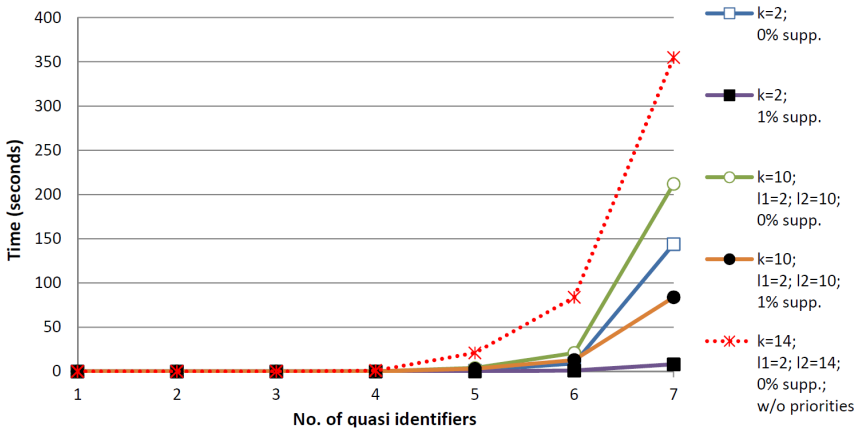

Fig. 4. Search algorithm complexity (time)

tions with a low number of partitions (and therefore big average partition size) are mostly bound to a high information loss, they are tested very late. As a consequence, the number of visited nodes is very high.

Two of the lines are especially interesting: the green one and the orange one. The green line deviates constantly until $k = 10$ at 1,255 nodes and then hits the maximal number of nodes 2,160 at the point $k = 14$. This means that in the first half, some particular outlier tuples are the bottleneck and force a high generalization, which with a 1% suppression would be remarkably lower (see the light blue line). The maximal possible ℓ value (14) is the bottleneck in the second half ($k \geq 14$) that causes a maximal table generalization and thus useless results. The orange line suffers from the same problem with outlier tuples as the first half of the green line. This chart confirms again that tuple suppression is essential and that it improves the quality of results tremendously and much more than any k adjustment could.

The experiment 6 shows similar results as 5 did, but focuses on anonymization with priorities (use of the cost function) vs. anonymization without priorities.

If we focus on green and orange line (only k-anonymity) or blue and red line (added ℓ-diversity), we notice that anonymizations without priorities visited a significantly higher amount of nodes to find a satisfying solution. The explanation for this is the lowest priority of *age* defined in anonymizations with priorities. Analysis of resulting generalization levels showed that *age* must be mostly generalized to the maximal level, to satisfy the privacy constraints. This is logical, since exactly *age* has far the most distinct values (see domain size in table 3). If we recapitulate how the algorithm with the cost function and the one without work, the big node number difference is clear. Best-first search generalizes *age* to its maximum first, since this is the cheapest move, and then starts to generalize the other attributes. A breadth-first search algorithm generalizes all

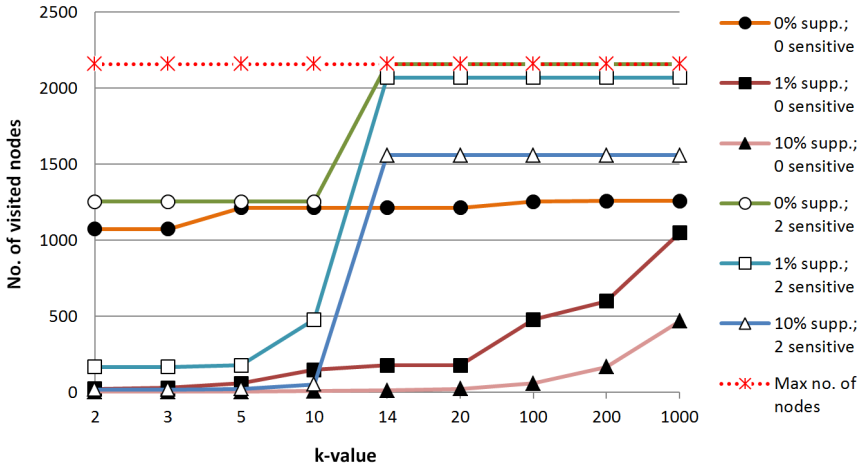

Fig. 5. Impact of k-value and suppression on the number of visited nodes (7 quasi identifiers)

attributes equally one by one (searches one tree level by another) and so visits many more nodes until it finds a solution (sometimes the same one). If *age* would have the highest priority, ceteris paribus, an algorithm without priorities would be better off.

This experiment showed that priorities should be used wisely. Sometimes, it is better to assign a slightly lower priority to an attribute that is important if the domain size is so big that the attribute would be generalized to a high level anyway. In other words: priorities can be used to increase performance (sink the number of visited nodes), too. In such a scenario, priorities should correlate negatively with the domain size (the bigger the domain size, the smaller the priority).

The experiment 7 shows how the user-defined priorities impact the number of visited nodes. The pink line shows how the number of nodes increases with ℓ if six quasi identifiers are anonymized. The line is quite flat until $\ell = 12$ and then rises rapidly to the almost maximal number of visited nodes (lower dotted line). This phenomenon was already discussed in experiment 5. The lower blue line with black circle markers ($k = 14$), which shows that the 14-anonymous table is a priori 6-diverse, reveals another interesting finding about ℓ: some ℓ values have a stronger anonymization effect as the others have (which may have none at all). The flat parts of the curve imply that if the 14-anonymous table satisfies 7-diversity, it is already 9-diverse. And a 10-diverse table is also 12-diverse. In other words: anonymizations with $\ell = 1$ to 6 (as well as $\ell = 7$ to 9 and $\ell = 10$ to 12) provide the same results. The same observation regarding k was made in pre-experiments, therefore only some striking k-values were used

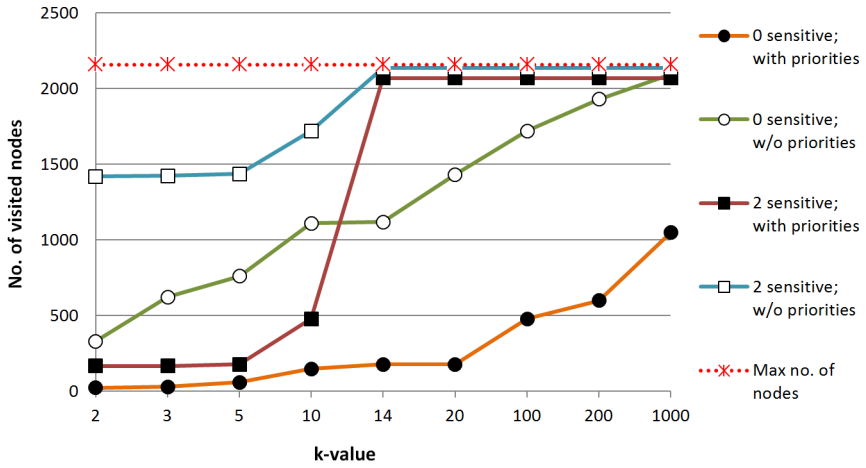

Fig. 6. Impact of user-defined priorities on the number of visited nodes (7 quasi identifiers, suppression 1%)

for further experiments. Both pairs (the pair with circular markers and the pair with triangular markers) have completely the same anonymization settings, but for the anonymizations with black markers the cost function (priority-weighted information loss) was used, whereas for the anonymizations with white markers it was not. The difference in number of visited nodes is tremendous. It is interesting that, until $\ell = 12$, a prioritized anonymization of a table with seven quasi identifiers (dark blue) performs better than a non-prioritized anonymization of a table with only six quasi identifiers (light blue).

This experiment proved that wisely used priorities can improve performance. Vice versa for an unartful use of them.

The experiments showed very clearly that the greatest performance and quality impact are the number of quasi identifiers and the maximal allowed suppression of outlier tuples. What we could not see in the experiments, is the impact of the rising number of sensitive attributes. Since the combinatorial effort grows with a rising number of sensitive attributes, the performance would suffer when the number of sensitive attributes would increase.

The impact of NULL values was evaluated in [4]. The experiments have shown that NULL value handling with the extended match leads to a lower information loss than no NULL value handling (basic match). Significant improvements can be reached with extended match especially on data with a high percentage of NULL values where a minor tuple suppression is allowed (e.g. 1% tuple suppression).

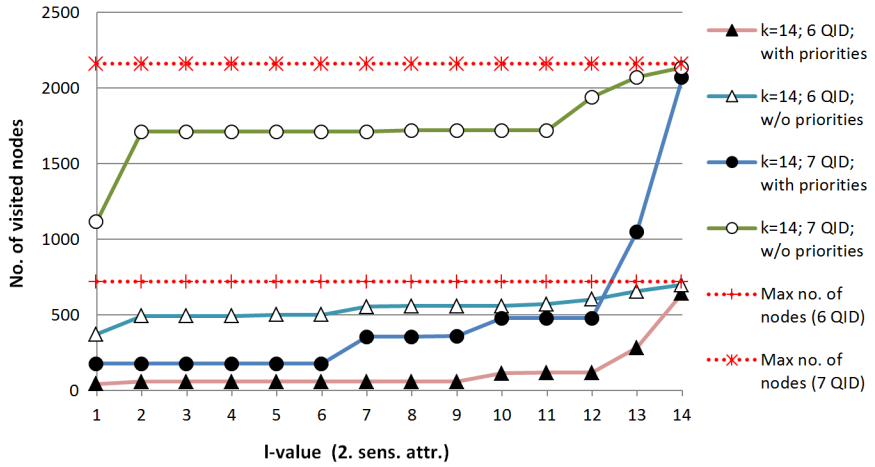

Fig. 7. Impact of user-defined priorities on the number of visited nodes (suppression 1%)

6 Conclusion

k-anyonymity and ℓ-diversity are well known and widespread concepts used to anonymize data, thus helping to protect the privacy of data owners. Surprisingly, both approaches simply ignore NULL values which frequently occur in micro-data, e.g. because some attributes are not applicable or because the data owner simply refused to reveal all information necessary.

In this technical report we first discussed our approach to deal with NULL values as presented in [4]. We implemented this approach as a web service written in Java. This tool, named ANON, will be published by TMF[1].

Furthermore we presented in detail both the algorithms which build the foundations of our implementation and the implementation architecure.

Finally, this technical reports presented the results of some experiments using real data (stemming from the U.S. census). We showed the impact of several parameters on both runtime performance and space consumption.

References

1. Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity. In *Proc. of the Int. Conf. on Database Theory (ICDT 2005)*, November 2005.

[1]TMF - Technologie- und Methodenplattform für die vernetzte medizinische Forschung e.V., Charlottenstraße 42, 10117 Berlin, http://www.tmf-ev.de

2. Gagan Aggarwal, Tomas Feder, Krishnaram Kenthapadi, Rajeev Motwani, Rina Panigrahy, Dilys Thomas, and An Zhu. Approximation algorithms for k-anonymity, 2005.
3. Roberto J. Bayardo and Rakesh Agrawal. Data privacy through optimal k-anonymization. In *Proc. of the 21st Int. Conf. on Data Engineering*, ICDE '05, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.
4. Margareta Ciglic, Johann Eder, and Christian Koncilia. *k*-anonymity of microdata with null values. In *Proc. of the 25th International Conference on Database and Expert Systems Applications - DEXA 2014*, 2014.
5. Lawrence H. Cox. Suppression methodology and statistical disclosure control. *Journal of the American Statistical Association*, 75(370):377–385, 1980.
6. J Eder, H Gottweis, and K Zatloukal. It solutions for privacy protection in biobanking. *Public Health Genomics*, 15(5):254–262, 2012.
7. Johann Eder, Claus Dabringer, Michaela Schicho, and Konrad Stark. Information systems for federated biobanks. In Abdelkader Hameurlain, Josef Küng, and Roland Wagner, editors, *Transactions on Large-Scale Data- and Knowledge-Centered Systems I*, volume 5740 of *Lecture Notes in Computer Science*, pages 156–190. Springer Berlin Heidelberg, 2009.
8. Johann Eder, Konrad Stark, M. Asslaber, P.M. Abuja, H. Gottweis, M. Trauner, H.J. Mischinger, W. Schippinger, A. Berghold, H. Denk, and K. Zatloukal. The genome austria tissue bank. *Pathobiology: journal of immunopathology, molecular, and cellular biology*, 74(4):251–8, 2007.
9. A. Frank and A. Asuncion. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2010.
10. Benjamin C. M. Fung, Ke Wang, Rui Chen, and Philip S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42(4):14:1–14:53, June 2010.
11. Benjamin C.M. Fung, Ke Wang, Ada Wai-Chee Fu, and Philip S. Yu. *Introduction to Privacy-Preserving Data Publishing: Concepts and Techniques.* Chapman & Hall/CRC, 1st edition, 2010.
12. George Gaskell, Herbert Gottweis, Johannes Starkbaum, Monica M Gerber, Jacqueline Broerse, Ursula Gottweis, Abbi Hobbs, Ilpo Helen, Maria Paschou, Karoliina Snell, and Alexandra Soulier. Publics and biobanks: Pan-european diversity and the challenge of responsible innovation. *Eur J Hum Genet*, 21(1):14–20, 2013.
13. Vijay S. Iyengar. Transforming data to satisfy privacy constraints. In *Proc. of the eighth ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, KDD '02, pages 279–288, New York, NY, USA, 2002. ACM.
14. Daniel Kifer and Johannes Gehrke. Injecting utility into anonymized datasets. In *Proc. of the 2006 ACM SIGMOD Int. Conf. on Management of data*, SIGMOD '06, pages 217–228, New York, NY, USA, 2006. ACM.
15. Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proc. of the 2005 ACM SIGMOD Int. Conf. on Management of data*, SIGMOD '05, pages 49–60, New York, NY, USA, 2005. ACM.
16. Kristen LeFevre, David J. DeWitt, and Raghu Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 49–60, New York, NY, USA, 2005. ACM.
17. Ninghui Li and Tiancheng Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *In Proc. of IEEE 23rd Int'l Conf. on Data Engineering (ICDE'07*, 2007.

18. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.
19. Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. L-diversity: Privacy beyond k-anonymity. *ACM Trans. Knowl. Discov. Data*, 1(1), March 2007.
20. G. J. Matthews and O. Harel. Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statist. Surv.*, 5:1–29, 2011.
21. Gregory J. Matthews and Ofer Harel. Data confidentiality: A review of methods for statistical disclosure limitation and methods for assessing privacy. *Statistics Surveys*, 5(0):1–29, 2011.
22. Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proc. of the 23rd ACM SIGMOD-SIGACT-SIGART symp. on Principles of database systems*, PODS '04, pages 223–228, New York, NY, USA, 2004. ACM.
23. Adam Meyerson and Ryan Williams. On the complexity of optimal k-anonymity. In *Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '04, pages 223–228, New York, NY, USA, 2004. ACM.
24. Aleksander Ohrn and Lucila Ohno-Machado. Using boolean reasoning to anonymize databases. *Artificial Intelligence in Medicine*, 15(3):235 – 254, 1999.
25. Hyoungmin Park and Kyuseok Shim. Approximate algorithms for k-anonymity. In *Proc. of the 2007 ACM SIGMOD Int. Conf. on Management of data*, SIGMOD '07, pages 67–78, New York, NY, USA, 2007. ACM.
26. Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Technical report, 1998.
27. Konrad Stark, Johann Eder, and Kurt Zatloukal. Priority-based k-anonymity accomplished by weighted generalisation structures. In *Proceedings of the 8th international conference on Data Warehousing and Knowledge Discovery*, DaWaK'06, pages 394–404, Berlin, Heidelberg, 2006. Springer-Verlag.
28. Xiaoxun Sun, Hua Wang, Jiuyong Li, and Traian Marius Truta. Enhanced p-sensitive k-anonymity models for privacy preserving data publishing. *Trans. Data Privacy*, 1(2):53–66, August 2008.
29. Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, October 2002.
30. Manolis Terrovitis, Nikos Mamoulis, and Panos Kalnis. Local and global recoding methods for anonymizing set-valued data. *The VLDB Journal*, 20(1):83–106, 2011.
31. Hongwei Tian and Weining Zhang. Extending l-diversity to generalize sensitive data. *Data & Knowledge Engineering*, 70(1):101 – 126, 2011.
32. H-Erich E. Wichmann, Klaus A. Kuhn, Melanie Waldenberger, Dominik Schmelcher, Simone Schuffenhauer, Thomas Meitinger, Sebastian H. Wurst, Gregor Lamla, Isabel Fortier, Paul R. Burton, Leena Peltonen, Markus Perola, Andres Metspalu, Peter Riegman, Ulf Landegren, Michael J. Taussig, Jan-Eric E. Litton, Martin N. Fransson, Johann Eder, Anne Cambon-Thomsen, Jasper Bovenberg, Georges Dagher, Gert-Jan J. van Ommen, Michael Griffith, Martin Yuille, and Kurt Zatloukal. Comprehensive catalog of european biobanks. *Nature biotechnology*, 29(9):795–797, September 2011.