2008

MultiRelational k-Anonymity

N. Ercan Nergiz

Chris Clifton
Purdue University, clifton@cs.purdue.edu

A. Erhan Nergiz

Report Number:

08-002

MULTIRELATIONAL K-ANONYMITY

M. Ercan Nergiz
Chris Clifton
A. Erhan Nergiz

CSD TR #08-002
January 2008

MultiRelational k-Anonymity

M. Ercan Nergiz, *Student Member, IEEE,*
Chris Clifton, *Senior Member, IEEE,* and
A. Erhan Nergiz

Abstract

k-Anonymity protects privacy by ensuring that data cannot be linked to a single individual. In a k-anonymous dataset, any identifying information occurs in at least k tuples. Much research has been done to modify a single table dataset to satisfy anonymity constraints. This paper extends the definitions of k-anonymity to multiple relations and shows that previously proposed methodologies either fail to protect privacy, or overly reduce the utility of the data, in a multiple relation setting. We also propose two new clustering algorithms to achieve multirelational anonymity. Experiments show the effectiveness of the approach in terms of utility and efficiency.

***Index Terms*— Privacy, Relational database, Security, integrity, and protection**

Note to reviewers: A preliminary version of this paper appeared as a five page poster paper at ICDE 2007: http://dx.doi.org/10.1109/ICDE.2007.369025 This submission includes additional discussion of the problems of single-table anonymization approaches, proofs of correctness, complexity discussion, a more efficient approximation evaluation, and empirical evaluation that did not appear in the ICDE poster paper.

I. Introduction

The tension between the value of using personal data for research, and concern over individual privacy, is ever-increasing. Simply removing uniquely identifying information (SSN, name) from data is not sufficient to prevent identification because partially identifying information (quasi-identifiers; age, sex, city ...) can still be mapped to individuals using publicly available knowledge [19]. Table I shows one such example where an attacker, by using a public dataset, can map the names of the students to the sensitive GPA information, even though the released private table does not disclose the names of the students. (E.g., a student with age "18", sex "M" and city "Lafayette" has GPA "2.34". Luke is the only person with these attributes in the public dataset.)

k-Anonymity[16] is one technique to protect against the linkage and identification of records. In a k-anonymous table, each distinct tuple in the projection over quasi-identifier attributes occurs at least k times. Private tables are k-anonymized by the use of generalizations and suppressions, with the result having two key properties:

- In the anonymous dataset, an individual can only be linked to a group of at least k private entities.
- Every tuple of the anonymous dataset correctly represents a unique tuple in the private dataset (There is no false or noisy information.)

Table I shows a 2-anonymization of the above mentioned private table. Given the 2-anonymized table, an attacker can at best link Luke into GPAs "3.72" and "2.34".

k-Anonymity does not enforce diversity on the sensitive information of equivalence classes (set of tuples with the same identifying attributes in k-anonymous dataset). This has lead to extended privacy definitions [6], [13]. However if all sensitive attributes in the private table are unique, k-anonymity ensures that linkage will only be possible to groups of k-distinct sensitive values.

To achieve k-anonymity in single-table datasets, numerous generalization (replacing data values with more general values) and suppression algorithms have been proposed [17], [7], [8], [10], [4], [11], [3], [5], [15]. These algorithms assume each private entity is stored as one row in a single attribute-value table. When information about a private entity is contained in multiple tables, and not easily represented in a single table, the existing definitions and algorithms are insufficient. In Section II, this paper extends the k-anonymity definitions to a multi-Relational setting; Section III discusses why multiR anonymity (multirelational k-anonymity) is a new problem that is not solved by previous k-anonymity algorithms.

Single dimensional k-anonymity algorithms were designed to specify generalization mappings (or complete suppression

This material is based upon work supported by the National Science Foundation under Grant No. 0428168.

of values) for data values in the dataset to optimize against a certain metric. Some of such algorithms used pruning methods to reduce the size of the search space for optimal k-anonymity [10], [4]. However in a multiR anonymity setting, the search space is much bigger and simple modifications won't be as efficient unless the original optimality is sacrificed by using other assumptions. In [15], [11], [5], it was shown that although not optimal, a multidimensional approach to k-anonymity can offer more flexibility in anonymizations. Among this family of algorithms, the clustering based approach is more suitable to the multiR setting due to the ease in explicit identification of the entity being protected (anonymized) in the dataset. In Section IV, protected entities and associated relations will be abstracted by trees and a modification of a previously proposed clustering algorithm will be presented to provide multiR anonymity on snowflake schemas. Section V will present experimental results evaluating the new approach in terms of precision and execution time.

II. MultiR Anonymity

We now define notations and k-anonymity for the multiR setting. Given a table T, $T[c][r]$ refers to the value of column c, row r of T. $T[c]$ is the projection of column c

Definition 1 (Person specific table): A table PT is said to be *person specific* w.r.t. some population U if and only if it contains a primary key attribute (or set of attributes) vip such that each value of vip uniquely corresponds to an individual in U.

Definition 2 (MultiR schema): A set of tables SU and a set of functional dependencies SF corresponds to a multiR schema if SU is a dependency preserving, lossless join decomposition w.r.t. SF and there exists one person specific table $PT \in SU$ where each row corresponds to an individual in population U. We say a database with such a schema has the transcript $MR(SF, U, PT, ST, vip)$, where vip is the unique identifier in PT and $ST = SU - \{PT\}$.

Table II shows an example for a multiR database with transcript $MR(SF, U, T_p, \{T_1, T_2\}, Sid)$ where SF={Sid $\rightarrow$ GPA, SCid $\rightarrow$ {Sid, Course, Grade} } and U is the set of students. The schema is in BCNF and dependency preserving.

The following quasi-identifier definition is a reformulation of the definition in [18].

Definition 3 (Quasi-identifier): Let $MR(SF, U, PT, \{T_1, \cdots, T_n\}, vip)$ be a multiR database, and $JT = PT \bowtie T_1 \bowtie \cdots \bowtie T_n$. Let $f_c : U \rightarrow JT$ and $f_g : JT \rightarrow U'$, where $U \subseteq U'$. A quasi-identifier of MR, written Q_{MR}, is a subset of attributes of JT where $\exists p_i \in U$ such that $f_g(f_c(p_i)[Q_{MR}]) = p_i$, and an adversary knows the values of Q_{MR} for p_i.

Informally a quasi-identifier for a schema is the set of attributes in JT that can be used to externally link or identify a given tuple in PT. In Table II, Course and Book attributes can be considered quasi-identifiers since colleagues of a student may know this information about their friend. The attributes

TABLE IV. Notations for a given database MR^i
table

vip^i	private entity attribute in MR^i
PT^i	the person specific table of MR^i (vip^i is the primary key)
ST^i	the set of all tables in MR^i excluding PT_i
SU^i	the set of all tables in MR^i
JT^i	join of all tables in MR^i
Q_{MR^i}	set of quasi identifier attributes
S_{MR^i}	set of sensitive attributes

GPA, Grade, Price are the sensitive attributes of the private entity Sid. An attacker knows the quasi-identifiers about an entity and tries to discover other (sensitive) information in the data. E.g., in Table II, we assume the attacker knows that some individual George in U takes the courses 'History' and 'Religion' and uses the text book 'American History' for the 'History' course. The attacker wants to discover George's (sensitive) GPA or his grade in the 'History' course. If the data is released as it is, even though George's name is hidden, the attacker can easily link George to student S4 and GPA '4.00' or SCid SC10 and grade '98'. We also have other join keys in Table II like the vip attribute Sid or SCid that are not part of the quasi-identifier set.

For the rest of the paper, we will use the notation given in Table IV. From now on, if not mentioned otherwise, we will use superscripts to name different multiR databases (e.g., $MR^1, MR^2, \cdots$). Superscript for other notations will show membership to the associated multiR database (e.g., vip^1 is vip of MR^1.). We will use superscript $*$ for multiR anonymizations. Subscripts will distinguish different elements of the same multiR database (e.g., $T_1^1, T_2^1 \in ST^1$ of MR^1).

Definition 4 (Structurally Equivalent): Two databases MR^1 and MR^2 have structurally equivalent schemas if and only if $vip^1 = vip^2$, PT^1 has the same set of attributes as PT^2, and there exist bijective mapping between the set of tables ST^1 and ST^2 such that tables mapped have the same set of attributes. Structurally equivalent schemas have the same functional dependencies, population, QI, sensitive and non-QI joining attribute sets.

The MultiR databases given in Tables II and III are an example of structural equivalence.

Definition 5 (k-anonymity for multiR databases): Let MR and MR^* be two multiR databases with the same set of QI Q_{MR} and set of sensitive attributes S_{MR}. We say MR^* is a k-anonymization of MR if and only if $\forall v(JT^*)$, (views on JT^*) the following properties hold:

1) *anonymized*: any query of the type $\Pi_{att}(v(JT^*))$ where $att \in S_{MR}$ returns either zero tuples or at least k (not necessarily distinct)[1] tuples,

[1] k-anonymity allows sensitive attribute values to be the same over the set of tuples with the same QI attributes. Other approaches like ℓ-diversity and t-closeness enforce constraints over the distribution of such groups of sensitive values.

TABLE I. An example public table (university registration database), private table (university alumni database) and an anonymization of the private table where $k = 2$
table

Public Dataset

Name	Age	Sex	City
Chris	19	M	Indianapolis
Luke	18	M	Lafayette
Padme	27	F	Lafayette
George	25	M	W. Lafayette

Private Dataset

Age	Sex	City	GPA
19	M	Indianapolis	3.72
18	M	Lafayette	2.34
27	F	Lafayette	3.12
25	M	W. Lafayette	4.00

Anonymized Dataset

Age	Sex	City	GPA
10-20	M	Indiana	3.72
10-20	M	Indiana	2.34
20-30	*	G. Lafayette	3.12
20-30	*	G. Lafayette	4.00

TABLE II. T_p:Student has GPA; T_1:Student takes courses; T_2:Books bought by student for course
table

Sid	GPA
S1	3.72
S2	2.34
S3	3.12
[redacted]	

T_p

SCid	Sid	Course	Grade
SC1	S1	Math	93
SC2	S1	Physics	91
SC3	S1	History	85
SC4	S2	CS	78
SC5	S2	Physics	62
SC6	S2	Religion	42
SC7	S3	History	85
SC8	S3	Religion	75
SC9	S3	Physics	77
[redacted]			

T_1

SCid	Book	Price
SC1	Discrete	$63
SC2	Calculus	$89
SC2	Dynamics	$42
SC3	Relg. H.	$33
SC4	Discrete	$65
SC5	Dynamics	$51
SC6	Yodaism	$38
SC7	Ottomans	$49
SC8	Yodaism	$39
SC9	Calculus	$84
[redacted]		

T_2

TABLE III. One anonymization of Table II where $k = 2$
table

Sid	GPA
S1	3.72
S2	2.34
S3	3.12
[redacted]	

T_p^*

SCid	Sid	Course	Grade
SC1	S1	*Science*	93
SC2	S1	Physics	91
SC3	S1	*Social*	85
SC4	S2	*Science*	78
SC5	S2	Physics	62
SC6	S2	*Social*	42
SC7	S3	History	85
SC8	S3	Religion	77
*	*	*	*
[redacted]			

T_1^*

SCid	Book	Price
SC1	Discrete	$63
SC2	Dynamics	$42
*	*	*
SC3	*Relg Book*	$33
SC4	Discrete	$65
SC5	Dynamics	$51
SC6	*Relg Book*	$38
SC7	*Hist Book*	$49
*	*	*
[redacted]		

T_2^*

2) *anonymized w.r.t. individuals*: any query of the type $\Pi_{vip}(v(JT^*))$ returns either zero tuples or at least k distinct tuples, and
3) *correct*: tuples in JT and JT^* can be ordered such that for all possible j, $JT^*[att][j]$ is equal to or some generalization of $JT[att][j]$ if $att \in Q_{MR}$ and $JT^*[att][j]$ is equal to $JT[att][j]$ if $att \in S_{MR}$

The part 'k not necessarily distinct tuples' in requirement 1 can be changed to 'k distinct tuples' if we assume all sensitive information in the MR is unique. MR and the k-anonymous MR^* need not be structurally equivalent, however, we will see that equivalence eases the anonymization process and can improve utility of the dataset.

The example in Table II is clearly not k-anonymous even for $k = 2$, as $|\Pi_{Sid}(\sigma_{Course='History' \wedge Book='Am.Hist'}(JT))| = |\{S4\}| = 1$. Table III shows a 2-anonymization of Table II using generalizations from the domain generalization hierarchies given in Figure 1; the same query on Table III returns no tuples.

Theorem 1: Let MR be a k-anonymous multiR database where $ST = \{T_1, \cdots, T_n\}$ and $k \geq 2$. Then for every vip value vp, there exist some $\ell \geq k-1$ distinct vip values $vp_1, \cdots$

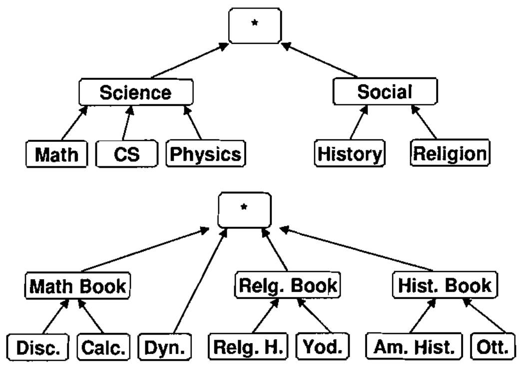

Fig. 1. Course, Book DGH structures

figure

vp_ℓ such that for every view v possible if $vp \in \Pi_{vip}(v(JT))$ then $vp_1, vp_2, \cdots vp_\ell \in \Pi_{vip}(v(JT))$. We say the set $S_{vp} = \{vp, vp_1, vp_2, \cdots vp_\ell\}$ is the equivalence class of vp and write $EC_{MR}(vp) = S_{vp}$.

PROOF. Suppose this is not the case and let the set of views $V_{vp} = \{v_i | vp \in \Pi_{vip}(v_i(JT))$. Since there are no common $k-1$ vip values (other than vp) over all views then we have $| \cap_{v_i \in V_{vp}} \Pi_{vip}(v_i(JT))| < k$. Constructing the view $v^\cap = \cap_{v_i \in V_{vp}} v_i$ gives $|\Pi_{vip}(v^\cap(JT))| \leq k$ and $vp \in \Pi_{vip}(v^\cap(JT))$, violating the k-anonymity constraint. This gives a contradiction. $\square$

The MR database in Table III, has two equivalence classes: {S1,S2} and {S3,S4}. (e.g., $EC_{MR}(S1) = \{S1, S2\}$)

Theorem 1 can be modified for only sensitive attributes if we have unique sensitive values. Every sensitive value s in the data belongs to a set $EC_{MR}(s)$ of at least k sensitive values such that if s is in a query result then every element in $EC_{MR}(s)$ is also in that query result. (e.g., in Table III, $EC_{MR}(3.72) = \{3.72, 2.34\}$)

The k-anonymity definition for a multiR database is not arbitrary. If an attacker faces the same set of private entities in every possible set of queries, it can only map its external knowledge to that set. Requirement 3 for k-anonymity prevents false information being included in the anonymization of the original database. (Otherwise there would be trivial solutions for k-anonymization such as replication of tuples. This requirement holds also for classical, single-table k-anonymity, although it was not included explicitly in its definition.) Note that the definitions and concepts given here subsume the definitions of single-table k-anonymity. In classical k-anonymity, we have one private table $PT(A_1, \cdots, A_n)$ without any dependencies corresponding to a population U. Since every tuple in PT belongs to an individual, we can add a unique identifier attribute to PT to form $PT_p(A_u, A_1, \cdots, A_n)$. PT_p becomes a person specific table with vip attribute A_u. In that case an anonymization for $MR(\{A_u \rightarrow \{A_1, \cdots, A_n\}\}, U, PT_p, \{\}, A_u)$ is also an anonymization for PT in terms of classical k-anonymity definitions.

We now define two operators that will be used in the following sections for multiR databases:

Definition 6 (Union): For structurally equivalent MR^1, MR^2 and $MR^\cup$, $MR^\cup \Leftarrow MR^1 \cup MR^2$ if and only if $PT^\cup = PT^1 \cup PT^2$, $(T_j^\cup \in ST^\cup) = (T_j^1 \in ST^1) \cup (T_j^2 \in ST^2)$.

Definition 7 (Concatenation): $MR^\| \Leftarrow MR^1 \| MR^2$ if and only if $PT^\| = PT^1$, $ST^\| = ST^1 \cup \{PT^2\} \cup ST^2$, and $vip^\| = vip^1$

Many different cost metrics were used in the literature [8], [4], [15], [9] to measure utility of anonymized datasets. We redefine two of these cost metrics, LM[8] and DM[4], for the multiR setting, and use them in our experiments. Different variations that may better fit to relational databases can be formalized. (Discussion on such a formulation is beyond the scope of this paper.) Algorithms in the coming sections are independent of the cost metric being used and discussions apply no matter what cost metric is being used.

Definition 8 (LM): $f(v)$ be a function that given a categorical [continuous] data cell value v returns the number of distinct values [value interval +1] that cell value stands for, and $g(att)$ be a function that returns the number of distinct values [value range +1] in from the domain of a given categorical [continuous] attribute att. Assuming $g(att) > 1$, the *general loss* metric for a multiR database MR^*

$$LM(MR^*) = \frac{\sum_{T \in SU^*} \sum_{qi \in QI_T} \sum_{j=1}^{|T|} \frac{f(T[qi][j]) - 1}{g(qi) - 1}}{\sum_{T \in SU^*} |T| \cdot |QI_T|}$$

LM metric can be defined on individual data cells. It penalizes the value of each data cell in the anonymized dataset depending on how general it is (how many leaves are below it on the DGH tree). (e.g., LM("Science") = $\frac{f(\text{"Science"})-1}{g(\text{"Course"})-1} = \frac{3-1}{5-1}$) LM for the multiR dataset normalizes the total cost to get a number between 0 and 1.

Definition 9 (DM): Let MR^* be an anonymization of MR and let $G_{MR^*}(vp)$ be the set of vips in MR^* indistinguishable from a given vip $vp \in MR$. Then

$$DM(MR^*) = \sum_{vp \in MR} |G_{MR^*}(vp)|$$

As in the LM metric, smaller the number returned by DM metric, better the anonymization.

III. Single Table Algorithms for MultiR Anonymity

We now explore some obvious approaches to achieving multiR anonymity using single table k-anonymity algorithms. The main idea is to convert the multiR database into one or more single tables and anonymize these. For each approach, we describe why it does not give satisfactory results; the

TABLE V. The universal table for T_p and T_1 along with 2 anonymizations of it where $k = 2$

table

Sid	Course	GPA
S1	Math	3.72
S1	Physics	3.72
S1	History	3.72
S2	CS	2.34
S2	Physics	2.34
S2	Religion	2.34
S3	History	3.12
S3	Religion	3.12
S3	Physics	3.12
S4	History	4.00
S4	Religion	4.00

JT

Sid	Course	GPA
S1	Science	3.72
S1	Science	3.72
S1	History	3.72
S2	Science	2.34
S2	Physics	2.34
S2	Religion	2.34
S3	History	3.12
S3	Religion	3.12
S3	Physics	3.12
S4	History	4.00
S4	Religion	4.00

AT_1

Sid	Course	GPA
S1	Science	3.72
S1	Science	3.72
{S1,S4}	History	3.72
*	*	2.34
{S2,S3}	Physics	2.34
{S2,S4}	Religion	2.34
S3	Social	3.12
S3	Social	3.12
{S2,S3}	Physics	3.12
{S1,S4}	History	4.00
{S2,S4}	Religion	4.00

AT_2

TABLE VI. Local anonymizations for T_p and T_1 where $k = 2$

table

Sid	GPA
S1	3.72
S2	2.34
S3	3.12
S4	4.00

T_p^1

Sid	GPA
{S1,S2}	3.72
{S1,S2}	2.34
{S3,S4}	3.12
{S3,S4}	4.00

T_p^2

Sid	Course
S1	Science
S1	Science
S1	History
S2	Science
S2	Physics
S2	Religion
S3	History
S3	Religion
S3	Physics
S4	History
S4	Religion

T_1^1

Sid	Course	Grade
S1	Science	93
S1	Science	91
{S1,S4}	History	85
*	*	78
{S2,S3}	Physics	62
{S2,S4}	Religion	42
S3	Social	85
S3	Social	75
{S2,S3}	Physics	77
{S1,S4}	History	98
{S2,S4}	Religion	96

T_1^2

Sid	Course
S1	Science
S1	Physics
S1	Social
S2	Science
S2	Physics
S2	Social
S3	History
S3	Religion
*	*
S4	History
S4	Religion

T_1^3

TABLE VII. Bitmap version of MR without some of the sensitive attributes and its 2-anonymization, attribute T in each course shows whether the student has taken that course or not. This reduces the info loss in the anonymization to some degree

table

Sid	Math		Physics			CS		History				Religion		GPA
	T	Di	T	Ca	Dyn	T	Di	T	RH	Ot	AH	T	Yo	
S1	1	1	1	1	1	0	0	1	1	0	0	0	0	3.72
S2	0	0	1	0	1	1	1	0	0	0	0	1	1	2.34
S3	0	0	1	1	0	0	0	1	0	1	0	1	1	3.12
S4	0	0	0	0	0	0	0	1	0	0	1	1	0	4.00

S1	*	*	1	*	1	*	*	*	*	0	0	*	*	3.72
S2	*	*	1	*	1	*	*	*	*	0	0	*	*	2.34
S3	0	0	*	*	0	0	0	1	0	*	*	1	*	3.12
S4	0	0	*	*	0	0	0	1	0	*	*	1	*	4.00

insights are useful in understanding the algorithm we will give in Section IV.

A. Universal Anonymization

One solution might be to construct the universal relation from the multiR database and run a single-table anonymization algorithm on this relation. Table JT in Table V shows the universal table for the database $MR(SF, U, T_p, \{T_1\}, Sid)$. (the attribute $SCid$ is removed but this does not affect the discussion.) To run an anonymity algorithm, we need to identify the attributes that need to be modified. We have two choices at this point. The first approach is to modify only the quasi-identifier attributes (attribute Course in JT) leaving the others untouched. Dataset AT_1 in Table V is one possible 2-anonymization of JT. However, we see that AT_1 obviously does not provide anonymity when an attacker knows all or some of the courses taken by a student. E.g., if an attacker knows that Chris is taking History, Math and Physics, then it will map Chris to $S1$ since $S1$ is the only one taking two science courses and a history course.

A second approach would be to modify join keys (NDGH generalizations [15]) along with the quasi-identifiers (e.g., attributes Course and Sid in JT). Dataset AT_2 in Table V is such a 2-anonymization of JT, but still fails to satisfy privacy constraints.

The main reason anonymization of a universal relation fails is that multiple tuples belong to a single person and the anonymization process does not take this into account. It becomes possible that tuples belonging to the same entity are anonymized with each other, making the relation "k-anonymous" but failing to protect individual identity. One way of resolving this would be to suppress all the data in the joining attributes (e.g., Sid). But in that case, the dataset would lose its relational structure and the valuable information in the 1-N or N-N relations (e.g., the information that a student taking Math, Physics and History has GPA 3.72 would be lost). This universal approach also suffers from inference channels due to the redundancy in representation when the adversary knows functional dependencies for the schema, e.g., in AT_2, given $Sid \rightarrow GPA$ holds, the attacker will discover the third tuple is actually Sid $S1$ since the first two tuples imply the student with GPA 2.71 is $S1$. A related work [21] worth mentioning here was on checking k-anonymity on views over a universal dataset. The work was not based on table generalizations and did not propose a k-anonymization algorithm to create anonymous views.

B. Local Anonymization

Another way to anonymize the dataset would be to k-anonymize each table independently. The most basic way of doing that is shown in T_p^1 and T_1^1 of Table VI. This set of tables suffers from the same problems mentioned in Section III-A (e.g., disclosure of Chris's GPA.)

A second approach again would be to use NDGH generalizations on non-QI join keys as shown in T_p^2 and T_1^2. In this case, for this particular MR database, GPA information seems to be 2-anonymous. However, sensitive Grade information is not protected. The attacker will still be able to map S1 to Chris and learn that he has received "93" and "91" in two science courses (although not which course each score belongs to.) This is a violation of anonymity requirement 2, since Chris is not anonymous with respect to another student. Another downside of the approach is that modifying join keys introduces many incorrect join paths, decreasing the usability of the data.

The main reason why local anonymizations fail is that use of independent and arbitrary mappings for generalization of one table can create inference channels with respect to mappings used by other tables. A multiR anonymity algorithm should use consistent mappings throughout datasets (e.g., by Theorem 1; if $S1$ and $S2$ are anonymized with each other in one table, their courses should also be anonymized with each other in the other table.) Tables T_p^2 and T_1^3 show a valid 2-anonymization that enforces consistent mapping. Anonymization should also decide which mapping to use for anonymization. Clearly a multiR anonymity algorithm needs to view data globally to come up with close mappings between private entities while maintaining precision and usefulness of the output data. The multiR anonymity algorithm given in Section IV will take all these observations into account and give global decisions for anonymization mappings.

C. Bitmap Anonymization

Some multiR databases can be converted to a boolean vector "bitmap" format with every private entity as a single row, and distinct attributes used to reflect different values.

Bitmap conversion is done by assigning the value "1" for attributes that the private entity *possess* in the MR database. Handling the other attributes that the entity does not possess is done differently for different types of MR databases. In *complete* databases, non-existing tuples in the db (*negative tuples*) implies that the individual does not possess the corresponding attribute. Thus non-existent tuples also constitute in the information content of the database. (e.g., University Registration Database, Voters Database, $\cdots$ In T_1 of Table II, S1 taking "Religion" course is missing implying Chris definitely did not take the "Religion" course.) In bitmap versions of complete databases, "0" is used for non-existent attributes of the entities. On the other hand, in *incomplete* databases, negative tuples imply uncertainty and they do not add into the information content. (e.g., hospital databases, business databases that share customers, $\cdots$ Having a patient not having a particular disease in a hospital database does not necessarily imply that patient did not have the disease. It is always possible that full records of a patient are contained in multiple hospitals.) In bitmap versions of incomplete databases, value "*" is used for non-existent attributes of the entities to express uncertainty.

Table VII shows the bitmap version of the complete MR database given in Table II and its 2-anonymization. Classical k-anonymity algorithms can be run on such datasets. The anonymized data will then satisfy both multiR anonymity requirements for certain types of relations, however:

1) Not every multiR database is bitmap convertible. Schemas containing tables that map one entity to another entity an arbitrary number of times cannot be converted to bitmap format without information loss. (E.g., a student taking n different Physic classes where n is arbitrarily large cannot be readily expressed. This is a serious drawback for datasets that are updated frequently. Updates on certain individuals can trigger changes in the schema of the anonymized dataset.)
2) For incomplete databases, anonymization would only be through suppression, as generalizing "S1 is taking a Math course and S2 is taking a CS course" into "S1 and S2 are both taking a Science course" would correspond to merging columns in the schema rather than generalization of data. So anonymizations cannot take advantage of user supplied generalization hierarchies or total ordering assumptions for the attribute domains (for the sake of both utilization and incorporating domain knowledge).
3) For complete databases, anonymizations would additionally preserve common negative information (e.g., "S3 is not taking a CS course and S4 is not taking a CS course", anonymization would preserve "neither S3 nor S4 is taking a CS course") However it is still impossible to incorporate domain knowledge through generalization hierarchies or total ordering assumptions. (e.g., generalizing a student taking "CS" with another student taking "Math" is as costly as generalizing two students taking "CS" and "Religion" respectively, even though the former could be a better generalization.)
4) Suppression in the bitmap setting removes certainty about the number of tuples corresponding to a given entity. (e.g., "S1 is taking a Math course and S2 is taking a CS course" could safely be generalized into "S1 and S2 are both taking at least one ("Science") course". Bitmap anonymization would imply "S1 and S2 are taking two courses in total".)
5) Bitmap anonymizations do not consider possible similarities of two private entities in the tail of a nested relation. (E.g., in the multiR database in Table II, S1 is taking a Math course, buys the Discrete book for the course and S2 is taking a CS course and buys the same book. Given that course information is generalized (or suppressed), the book information can safely be preserved without violating privacy. Bitmap anonymization would not retain *only* the book information.)
6) Conversion to bitmap format produces datasets of high dimensionality. Since distribution of produced data points are skewed over the whole possible space, this does not introduce further problems regarding the curse of dimensionality. However, k-anonymity algorithms do not take into account the existence of 'invalid points' (e.g., a point with Math-T:0, Math-Di:1 would be an invalid point implying student has not taken 'Math' but used the 'Discrete' book for the 'Math' course. Heuristics would need to be used that would ignore invalid points to speed up the anonymization.
7) Most real world data is stored as relational tables rather than bitmap tables. Conversion to such a bitmap costs additional execution time and storage, not to mention the cost of converting applications designed for the original schema.
8) Many real world relational databases contain correlations within relations and this may make certain heuristics for improving efficiency possible. (e.g., a student taking a 'science' course is more likely to buy a 'science' or 'math' book than a 'religion' book. It is possible to design fast and reasonably precise algorithms that decide anonymizations only on courses without considering book information.) It may be difficult to exploit such correlations without considering the structure of the data. A single table k-anonymity algorithm on a bitmap database will be unaware of the underlying structure and thus the correlation.

IV. Clustering-based MultiR Anonymity

We now develop a multiR anonymity algorithm that overcomes the shortcomings of the approaches described in the previous section, although it places certain (reasonable) restrictions on the schemas supported. Algorithms for arbitrary schemas are left as future work.

A. Assumptions and Properties

We aim to preserve certain properties of the database, and in doing so accept certain limitations on the databases that can be anonymized by our algorithm. These properties and assumptions are given here.

Schema Preservation: The schemas of the input database MR and the k-anonymous output MR^* will be structurally equivalent (Definition 4).

Dependency Preservation: The anonymized database preserves functional dependencies of the original database, so that:

1) the semantics of the data are better preserved, and
2) inference attacks, by an adversary who knows a functional dependency that *fails* to hold in the anonymized data, are prevented.

We require that the schema be normalized to enforce dependencies; this obviates the need to provide dependencies separately as input to the anonymization algorithm.

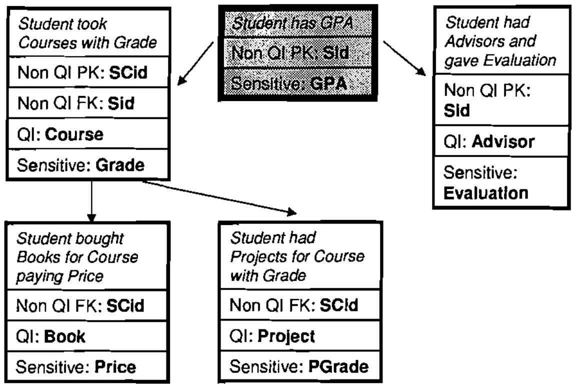

Fig. 2. Schema graph

figure

Snowflake Schema: The algorithm we present is limited to schemas satisfying the following constraints:

1) No connection keys (primary/foreign keys) between tables in MR are quasi-identifiers. (It is possible to replace such quasi-identifiers with non-identifying keys to preserve connections.)
2) Every table in ST contains only one foreign key. Table PT does not contain a foreign key.
3) We say a table T_2 belongs to the family of T_1 and write $T_2 \in F(T_1)$ if T_2 has a foreign key attribute which is a primary key attribute either in T_1 or in another family member of T_1. We restrict ourselves to schemas with $F(PT) = ST$.

Schemas with these constraints are similar to snowflake relations where the fact table is the table PT (see Figure 2), although we do support one to many relationships between PT and other tables. Any table in the schema can contain sensitive attributes; anonymity constraint 1 will hold for all of them. This family of schemas is expressive enough for many database applications (XML, some spatio-temporal databases, data warehouses, ...)

Join Key Atomicity: The algorithm presented in the next section will preserve the atomicity of join keys. (The assumption that join keys are not quasi-identifiers makes it possible to follow this approach in all cases.) This ensures one true join path as opposed to multiple paths (as in $\{T_p^2, T_1^2\}$ of Table VI) in each connection and improves utility of the anonymization (a query on the anonymized dataset is "true", in the sense that the result is a generalization of the result on the underlying dataset.)

B. MultIRelAtional CLustEring (MiRaCle)[8] Anonymization Algorithm

We now present a MiRaCle anonymization algorithm that anonymizes a given multiR database under the assumptions given in the previous section. We first give a higher level description of the algorithm to make the formal explanation easy to follow.

1) Informal Description: MiRaCle is a clustering-based anonymity algorithm; any distance-based clustering k-anonymity algorithm [5], [15], [1] can be used as a basic skeleton for MiRaCle anonymizations. The main observation is that all clustering based anonymity algorithms make use of two basic operations on private entities: anonymization and calculation of the distance between two entities. The latter can be generally defined as the cost of the anonymization of two entities. As an example basic skeleton, in the next section, we present a trivial modification of CDGH clustering algorithm [15] for MiRaCle. Here we turn our attention to the real question: *How to anonymize two entities?*

The assumptions given in the previous section enables us to abstract entities of a multiR databases as trees where each level of a given entity tree corresponds to levels of the nested relation for a particular vip entity. (Figure 3 gives an example.) The challenge is to anonymize two trees of similar structure with respect to each other.

Algorithm 1 anonymize($tree(s_1)$, $tree(s_2)$)

Require: For a tree node s; $tree(s)$ returns the tree rooted from s and v_s returns the QI attribute values associated with node s. For two values of the same domain v_1 and v_2, $gen(v_1, v_2)$ returns the lowest cost generalization of v_1 and v_2 w.r.t. a dgh.

$v_{c_1}, v_{c_2} = gen(v_{c_1}, v_{c_2})$
let C_1 be the set of child nodes of node s_1
let C_2 be the set of child nodes of node s_2
find a low cost pairing of nodes in C_1 and C_2
for all matching pairs of nodes $(c_1 \in C_1, c_2 \in C_2)$ **do**
  anonymize($tree(c_1)$, $tree(c_2)$)
for all nodes $c \in (C_1 \cup C_2)$ unmatched **do**
  suppress every value in nodes of $tree(c)$

Algorithm 1 shows how to anonymize two entity trees. Anonymization occurs top-down. First QI attributes for tree roots are anonymized with each other. Each tree root has a set of child nodes. (In Figure 3, children of $S1$ and $S2$: C_1={"Math", "Physics", "History"}, C_2={"CS", "Physics", "Religion"}.) The algorithm chooses pairings of nodes between these sets to minimize the local cost in the current level or the overall cost of the anonymized trees. (In Figure 3, "Math" is paired with "CS", "Physics" with "Physics", and "History" with "Religion", producing the set of nodes {"Science", "Physics", "Social"} which is the least costly set in terms of the cost metric used (e.g., LM.) Since each pair is composed of two trees to be anonymized and function is

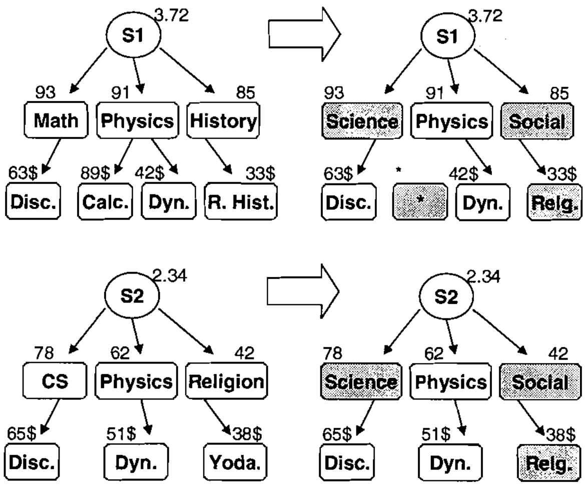

Fig. 3. Anonymization of students S1 and S2 from the example MR database in Table II

figure

called on the subtrees. (In Figure 3, a second call is made on ($tree$("Math"), $tree$("CS")). "Math" and "CS" values are changed to "Science" as a result of the second call . Unpaired nodes are suppressed (e.g., node "Calc.")

2) Formal Description: We first show in Algorithm 2 how to modify the CDGH clustering algorithm [15] to anonymize a given multiR database. Each cluster has a representative that holds the anonymization of the entities it contains. For each vip value v, the algorithm finds, in line 5, a suitable cluster to put v into. Suitability is measured by a distance function $dist$ which we will define shortly. If there is no suitable cluster, in line 7. v defines a new one. Then in line 9, the cluster representative of the closest cluster is updated to be the anonymization of v and the former representative by calling the function *anon.* When a cluster is full, the identifying information in the tuples in the cluster (including tuples linked to in other tables) is replaced with the cluster representative; these generalized tuples are placed into the anonymized database and the cluster is deleted. In lines 13-20, leftover clusters are combined. Leftover tuples in the last cluster ($< k$) are suppressed.

As also mentioned in the previous section, the real challenge is to define the distance between the two points (e.g., private entities such as students). If we know how to produce anonymizations of two points with respect to each other, we can derive the distance between them by calculating the cost of their anonymization w.r.t. any precision/cost metric. Here are formal details regarding how MiRaCle defines the anonymization and distance functions between two private entities (vips) $v_1 \in MR^1$ and $v_2 \in MR^2$:

Algorithm 2 MiRaCle($MR, k, th, climit, anon, dist, cost$)

Require: An input database MR with $ST = \{T_1, \cdots, T_n\}$, k constraint, a threshold value th, a cluster limit $climit$;
an anonymization function $anon$ that can anonymize two private entities;
a distance function $dist$ that can calculate the distance of two private entities;
a cost metric function $cost$ defined over anonymized MR databases;
We begin with an empty set of clusters C. vip v_{c_i} is the cluster representative of cluster c_i, MR_{c_i} is the database that contains v_{c_i} and EC_{c_i} holds the set of private entities in c_i .

Ensure: MR^* is a k-anonymization of MR

1: $MR^* \leftarrow null$
2: **for all** vip value v_j in PT **do**
3: **if** C is empty **then**
4: go to line 7
5: find i s.t. $d_i = dist(v_j, v_{c_i}, MR, MR_{c_i})$ is minimum
6: **if** $(d_i > th) \wedge (|C| \leq climit)$ **then**
7: make a new cluster c_{new}, set cluster representative $v_{c_{new}} = v_j$, $MR_{c_{new}} = MR$, $C = C \cup \{c_{new}\}$, $EC_{c_i} = \{v_j\}$
8: go to step 2 to process the next vip in MR
9: $MR_{c_i} = anon(v_{c_i}, v_j, MR_{c_i}, MR)$.
10: $EC_{c_i} = EC_{c_i} \cup \{v_j\}$
11: **if** the number of elements in c_i becomes more than k **then**
12: $MR^* = MR^* \cup MR_{c_i}$; $C = C - c_i$ (remove c_i)
13: **for all** cluster c_i left in C **do**
14: find $j \neq i$ s.t. $d_i = dist(v_{c_j}, v_{c_i}, MR_{c_j}, MR_{c_i})$ is min.
15: $MR_{c_i} = anon(v_{c_i}, v_{c_j}, MR_{c_i}, MR_{c_j})$.
16: $EC_{c_i} = EC_{c_i} \cup EC_{c_j}$; $C = C - c_j$ (remove c_j)
17: **if** the number of elements in c_i becomes more than k **then**
18: $C = C - c_i$ (remove c_i); $MR^* = MR^* \cup MR_{c_i}$
19: **else**
20: go to line 14 to find another suitable j.
21: MR^* now contains only one vip v_i data for each equivalence class, add the anonymizations for other vips by using EC_{c_j} sets created in the process.
22: suppress the remaining vips in C and add to MR^*
23: return MR^*

$$anon(v_1, v_2, MR^1, MR^2) =$$
$$Anonymize(\sigma_{vip^1=v_1} PT^1, \sigma_{vip^2=v_2} PT^2, MR^1, MR^2)$$
$$dist(v_1, v_2, MR^1, MR^2) =$$
$$cost(anon(v_1, v_2, MR^1, MR^2))$$

For each entity in the input MR db, MiRaCle makes one call to function *Anonymize* per cluster representative. Since the number of cluster representative is bounded by the input parameter $climit$, MiRaCle calls *Anonymize* $O(climit \cdot |MR|)$ times. The efficiency of the algorithm depends on the efficiency of the *Anonymize* function.

Function "Anonymize(t^1, t^2, MR^1, MR^2)" produces an anonymization for two tuples $t^1 \in PT^1$ and $t^2 \in PT^2$. (t^i

Algorithm 3 Anonymize(t^1, t^2, MR^1, MR^2)

Require: Tuple t^i belongs to table PT^i. All MR^i are structurally equivalent, Function $gen(v_1, v_2)$ returns the common parent of values v_1, v_2 on the dgh structure of the associated domain.
Ensure: MR^* is an anonymization of t^1 and t^2
1: $T^* \Leftarrow NULL$
2: Let MR^* be a database with transcript $(.,.,T^*,\{\},vip^1)$
3: **for all** att_i of PT^1 **do**
4: **if** att_i is a QI attribute **then** {Just anonymize}
5: $T^*[att_i][1] \Leftarrow gen(t^1[att_i], t^2[att_i])$
6: **if** att_i is a non-QI non-key or a foreign key **then** {Copy}
7: $T^*[att_i][1] \Leftarrow t^1[att_i]$;
8: **if** att_i is a primary key for a join with another table **then** {Ensure anonymized across join}
9: **for all** pairs of tables T_k^1, T_k^2 in MR^1, MR^2 where att_i is a foreign key **do**
10: Let MR_k^j be the database with transcript $\{.,.,T_k^j, F(T_k^j), att_i\}$
11: $MR^* \Leftarrow MR^* \|$ AnonymizeSets ($\sigma_{att_i=t^1[att_i]}T_k^1$, $\sigma_{att_i=t^2[att_i]}T_k^2$, MR_k^1, MR_k^2)
12: $T^*[att_i][1] \Leftarrow t^1[att_i]$
13: return MR^*

Algorithm 4 AnonymizeSets($C^1 = \{t_1^1, t_2^1, \cdots t_m^1\}, C^2 = \{t_1^2, t_2^2, \cdots t_n^2\}, MR^1, MR^2$)

Require: Sets of tuples C^i belongs to tables PT^i. All MR^i are structurally equivalent. $1 \leq m \leq n$
Ensure: MR^* is a pairwise anonymization of C^1 and C^2
1: Let MR^* be an empty database, structurally eq. to MR^i.
2: **for all** $t_i^1 \in C^1$ **do**
3: **for all** $t_j^2 \in C^2$ **do**
4: $tempMR_j \Leftarrow Anonymize(t_i^1, t_j^2, MR^1, MR^2)$
5: $costMR_j \Leftarrow cost(tempMR_j)$
6: $minCostj \Leftarrow \arg\min_j costMR_j$
7: $MR^* \Leftarrow MR^* \cup tempMR_{minCostj}$
8: $C^2 \Leftarrow C^2 - t_{minCostj}$
9: Suppress rest of the tuples in C^2 and add them to PT^*
10: return MR^*

may be considered as a root node of a tree structure stored in database MR^i, e.g., Figure 3) The function classifies and processes each attribute one by one. Processing of primary key attributes is important since they serve as connections to other tables. Attribute evaluation can be summarized as follows:

- Lines 4-7: for non-key attributes and foreign key attributes, behave as in single table anonymity: anonymize QI attributes w.r.t. dgh structures, leave the rest (sensitive attributes and foreign keys) as they are.
- Lines 8-12: for a primary key attribute *att*, find all pairs of tables ($T_k^1 \in ST^1, T_k^2 \in ST^2$) where *att* is a foreign key. We will have two sets of tuples $C^1 = \{t_1^1, \cdots, t_n^1\}$ and $C^2 = \{t_1^2, \cdots, t_m^2\}$ in T_k^1 and T_k^2 respectively where each $t_i^1[att] = t^1[att]$ and each $t_i^2[att] = t^2[att]$. Call "anonymizeSets($C^1, C^2, ., ., .$)" to find suitable one-to-one matchings between t_i^1s and t_j^2s. Suitability of a given matching depends on the effect of the generalization on all of the connected tables (This is ensured by recursive calls to the anonymization function in Line 4.) Anonymize matched tuples with each other, suppressing any unmatched tuples.

Given sets of tuples C^1 and C^2, and assuming $n = |C^1| = |C^2|$ there are $O(n!)$ possible pairwise matchings. It is costly to search such a big space to find a cost optimal matching. Because of this, algorithm *anonymizeSets* uses the following matching heuristic. Each node in C^1 is matched optimally with a node in C^2 one by one. (e.g., t_1^1 is matched with a tuple in C^2, then t_2^1 is matched with another, $\cdots$) This way complexity reduces to $O(n^2)$ pairwise matchings.

The algorithm can use any incremental cost metric that can be defined on a database. For the experiments, we will use the LM metric defined in Section II.

Table III shows the output of MiRaCle on the *MR* input given in Table II for $k = 2$. *vip* S1 and S2, and *vip* S3 and S4 anonymized with each other. Figure 3 shows how S1 and S2 are anonymized. The algorithm first ensures the tuples are anonymous w.r.t. QI attributes. Since T_p does not contain any QI attributes, no change is done (the root nodes in Figure 3). However, the primary key of T_p, *Sid*, occurs in T_1 as a foreign key, so algorithm AnonymizeSets is called on the sets of tuples $\sigma_{sid=\text{"}S1\text{"}}T_1$ and $\sigma_{sid=\text{"}S2\text{"}}T_1$ (the nodes on the second level of the trees). A one-to-one matching of tuples is done according to how costly the anonymization of the matched tuples will be. Anonymization in this level also takes into account table T_2 (Books table), since T_2 and T_1 share *SCid* as a joining key. First, the "Math" node is matched with the "CS" node since they can be anonymized as "Science" and they have a common node in the third level (in table T_2). The "Physics" node is matched with "Physics", the anonymization here triggers a call of AnonymizeSets on the sets of nodes {"Calc", "Dyn"} and {"Dyn"}. Node "Dyn" is matched with node "Dyn". No match is found for the node "Calc" so it is suppressed. The last nodes in the second level are anonymized similarly.

If we take the function *gen* as the basic operation, function *anonymize* (and thus the algorithm MiRaCle) turns out to be expensive. Assuming $n = |C^1| = |C^2|$, for every call to $anonymizeSets(C^1, C^2, ., .)$, $O(n^2)$ generalizations are performed. Note that the *anonymize* function (thus function *anonymizeSets*) is recursively called for every level in the relation (roughly speaking for every table in the MR database). Given that we have ℓ levels (tables) in MR, complexity function is defined as $f(\ell) = n^2 \cdot f(\ell - 1)$. This gives us a complexity of $O(n^{2\ell})$ for function *anonymize*. So MiRaCle is an O($climit \cdot |MR| \cdot n^{2\ell}$) algorithm.

C. MiRaCle Extension: MiRaCleX

As mentioned in the previous sections, a multiR anonymization algorithm can make use of the relational structure of the database to come up with more efficient heuristics. We present one example of such a heuristic in this section.

The MiRaCle anonymization process given in Section IV-B.2 considers the whole sibling subtrees when deciding on a suitable matching of sibling nodes. (in other words, *subtree matching* is done rather than *node matching*.) This is an effective way of achieving an anonymization with maximum precision. However, it is costly in terms of execution time since the *Anonymize* function has to be called for each potentially matched subtree pair (even for pairs that are not matched at the end of the anonymization process).

MiRaCle extension, MiRaCleX, makes use of the following observation: *If QI values for two root nodes are similar, then QI values for their children are likely to be similar too.* (If two students are both taking "Math" course, it is probable that they are both using a "Math" book.) This observation can be generalized for most relational databases. (The tail of the relations is correlated with the root of the relation.) An algorithm may produce anonymizations with reasonable precision much faster by just looking at the QI attribute similarities of the upper level nodes of the relation and not considering lower level nodes. Given this, pairing of sibling nodes in the *AnonymizeSets* function of MiRaCleX can be rewritten as in Algorithm 5. By this, the recursive call to the *Anonymize* function is moved outside of the innermost loop and the complexity function for function *anonymize* becomes $f(\ell) = n \cdot f(\ell - 1) + n^2$. This gives us a complexity of $O(n^{\ell+1})$ for function *anonymize*. So MiRaCleX is an O($climit \cdot |MR| \cdot n^{\ell+1}$) algorithm.

In Figure 3, to find a matching between {"Math","Physics[1]","History"} and {"CS","Physics[2]","Religion"} in the second level, MiRaCleX *Anonymize* function only considers QI attributes in the Course table T_1, ignoring information in the Books table T_2. Once matching is done on the second level (e.g., "Physics[1]" to "Physics[2]"), QI attributes in the Books table specify the matching on the third level (e.g., a matching between {"Calc","Dyn"} and {"Dyn"}).

D. Proof of k-Anonymity for MiRaCle Anonymization Algorithm

Now we prove that MiRaCle produces k-anonymous databases[2]. Since the algorithm preserves the structure of the data and all changes are based on either generalizations or suppressions, the third requirement for k-anonymity trivially holds. The following theorems prove the first requirement (sensitive information protection). The proof for the second requirement is similar. Since k-anonymity ensures total protection against sensitive information disclosure only when sensitive information is unique for every tuple, throughout the proof, we assume such constraint is enforced in the dataset and prove sensitive information is k-anonymous in the output dataset. We assume the schemas satisfy the assumptions given in Section IV-A.

Algorithm 5 AnonymizeSetsX($C^1 = \{t^1_1, t^1_2, \cdots t^1_m\}, C^2 = \{t^2_1, t^2_2, \cdots t^2_n\}, MR^1, MR^2$)

Require: Sets of tuples C^i belongs to tables PT^i. All MR^i are structurally equivalent. $1 \leq m \leq n$
Ensure: MR^* is a pairwise anonymization of C^1 and C^2

```
let MR* be an empty database, structurally eq. to MR^i.
for all t^1_i ∈ C^1 do
  for all t^2_j ∈ C^2 do
    for all attribute att of t^1_i do
      if att is a QI attribute then
        t*_j[att] ⇐ gen(t^1_i[att], t^2_j[att])
      else
        t*_j[att] ⇐ t^1_i[att]
  minCostj ⇐ arg min_j cost(t*_j)
  tempMR ⇐ Anonymize(t^1_i, t^2_minCostj, MR^1, MR^2)
  MR* ⇐ MR* ∪ tempMR
  C^2 ⇐ C^2 − t^2_minCostj
suppress rest of the tuples in C^2 and add them to PT*
return MR*
```

We start by showing that anonymization of two private entities is correctly carried out by the function *Anonymize*. The function *Anonymize* given in Algorithm 3 produces one representation of the anonymization as opposed to multiple copies of it. For each equivalence class, copies are produced from the representation at the end of MiRaCle given in Algorithm 2. It is trivial to modify the function *Anonymize* to output the necessary copies. The proofs below will assume copies exist in the *Anonymize* output. Since the algorithm structure is recursive, we first prove the base case:

Lemma 2: Let MR^1 and MR^2 have structurally equivalent schemas with $ST^i = \{\}$. Let t^i be a tuple in PT^i. Then function "Anonymize(t^1, t^2, MR^1, MR^2)" produces a 2-anonymization for the tuples t^1 and t^2.

PROOF. Since there are no tables connected to PT^i, *Anonymize* only applies basic generalizations to QI attributes of t^i as in the single table k-anonymization process. This ensures each QI in the two anonymized tuples is the same. Therefore any subset of the QI occurs in at least two tuples; with no links to other tables, 2-anonymity holds.[3] □

We now prove, in a bottom up fashion, the recursive step to prove that k-anonymity property is propagated through connected tables: If we take a set of k-anonymous databases, and add another k-anonymous table where the join keys for each set of private entities join (only) with an equivalence class in the table, and vice-versa, then the combined set of tables is k-anonymous.

Lemma 3: Let $MR^1, \cdots, MR^i, \cdots, MR^t$ be t structurally equivalent k-anonymous databases with set of sensitive attributes S, QI attributes $Q = \{qi_1, \cdots, qi_l\}$ and a common

[2]Discussion also applies for MiRaCleX

[3]The algorithm behaves exactly like CDGH anonymization algorithm [15] in this case.

vip attribute vip. Suppose PT^is contain a key pri. Let $EC_{MR^i}(pri')$ returns the set of pri values that belong to the equivalence class of the pri value pri' in MR^i. Also suppose for any value pri', $EC_{MR^a}(pri') = EC_{MR^b}(pri')$ if $pri' \in PT^a, PT^b$. That means equivalence classes of attribute pri are the same in all MR^i. Let $EC_{MR}(pri')$ return this *universal* equivalence class of pri'.

Let MR^{root} be another k-anonymous db with transcript $(.,.,T,\{\},pri)$. Suppose T has attributes $(pri, att_1, \cdots, att_m, sen_1, \cdots, sen_n)$. By definition pri is the primary key, att_is are QI attributes , and sen_js are sensitive attributes. (Note that T should be also k-anonymous.) and also suppose $EC_T(pri') = EC_{MR}(pri')$ for every possible pri'. Then $MR = MR^{root}||(\bigcup_i MR^i)$ is also k-anonymous.

As an example for Lemma 3, in Table III, MR^1=$\{.,.,\sigma_{Course=\text{"Science"}}T_1^*, \{\sigma_{SCid=SC1 \vee SCid=SC4}T_2^*\}, SCid\}$, MR^2=$\{.,.,\sigma_{Course=\text{"Physics"}}T_1^*, \{\sigma_{SCid=SC2 \vee SCid=SC5}T_2^*\}, SCid\}$. The pri attribute above corresponds to the attribute Sid and $MR^{root} = \{.,.,T_p^*,\{\},Sid\}$.

PROOF. Suppose this is not the case and there exists a query Q on the join JT where $0 < |\Pi_s(Q(JT))| < k$ for some sensitive s which is an attribute either in S or in table T. We will look at each case separately. First suppose $s \in S$ and some $s' \in \Pi_s(Q(JT))$. This implies that there exists at least one tuple $t(pri = p, att_{1\cdots m} = a_{1\cdots m}, vip = v, qi_{1\cdots\ell} = q_{1\cdots\ell}, s = s') \in JT$ (otherwise s' has no connection with T and we get a contradiction from the k-anonymity of the MR^i), and $(pri = p, att_{1\cdots m} = a_{1\cdots m}) \in T$. Now suppose s' occurs in MR^a $(1 \leq a \leq j)$ and $(vip = v, pri = p, s = s', qi_{1\cdots\ell} = q_{1\cdots\ell}) \in JT^a$. Since MR^a is k-anonymous, $(vip = v_j, pri = p_j, s = s_j, qi_{1\cdots\ell} = q_{1\cdots\ell}) \in JT^a$ also holds, for every $p_j \in EC_{MR}(p)$ and for distinct s_j. By the definition of T, if $(pri = p, att_{1\cdots m} = a_{1\cdots m}) \in T$, also $(pri = p_j, att_{1\cdots m} = a_{1\cdots m}) \in T$ holds for the same set of p_js. However, in that case $(pri = p_j, att_{1\cdots m} = a_{1\cdots m}, vip = v, qi_{1\cdots\ell} = q_{1\cdots\ell}, s = s_j) \in JT$. This means we have at least $k-1$ other s values with the same QI attributes as s'. (e.g., consider table T_p in Figure 3, p=S1 and one MR^a is the two generalization trees with $s = 93, 78$ respectively and both rooted from "Science" node with $EC_{T_p}(\text{S1}) = EC_{MR^a}(\text{S1})$={S1,S2}. As S1 is connected to one tree, S2 is connected to the other. This is true for all other MR^as: two MR dbs rooted from "Physics" and "Social" nodes respectively. It is impossible to distinguish S1 from S2 by using only QI attributes.) Then if $s' \in \Pi_s(Q(JT))$, $s_j \in \Pi_s(Q(JT))$ meaning $|\Pi_s(Q(JT))| \geq k$.

The proof is similar when s is an attribute from T. Suppose again $s' \in \Pi_s(Q(JT))$ and $(pri = p, att_{1\cdots m} = a_{1\cdots m}, vip = v, qi_{1\cdots\ell} = q_{1\cdots\ell}, s = s') \in JT$. In this case, p may occur in more than one MR^a but since equivalence class of p is the same in each of them, discussion is still valid. In this case, we have $(pri = p, att_{1\cdots m} = a_{1\cdots m}, s = s') \in T$ and $(vip = v, pri = p, qi_{1\cdots\ell} = q_{1\cdots\ell}) \in JT^a$. Since MR^a is k-anonymous, $(vip = v_j, pri = p_j, qi_{1\cdots\ell} = q_{1\cdots\ell}) \in JT^a$ also holds, for every $p_j \in EC_{MR}(p)$. By the definition of T, $(pri = p_j, s = s_j, att_{1\cdots m} = a_{1\cdots m}) \in T$ holds for same p_js and distinct s_j. Again we will have, $(vip = v_j, pri = p_j, att_{1\cdots m} = a_{1\cdots m}, qi_{1\cdots\ell} = q_{1\cdots\ell}, s = s_j) \in JT$ and $s_j \in \Pi_s(Q(JT))$. $\square$

Theorem 4: Let MR^1 and MR^2 have structurally equivalent schemas with $ST^i = \{T_1^i, \cdots, T_n^i\}$ and tuple $t^i \in PT^i$. Then function "Anonymize(t^1, t^2, MR^1, MR^2)" produces 2-anonymization for the tuples t^1 and t^2 in some MultiR db MR^*.

PROOF. Without loss of generality, suppose only T_1^is directly joins with PT^is. In Lines 4-7, the algorithm first generalizes t^1 and t^2 with each other. This provides 2-anonymity for t^1 and t^2 locally in PT^*. (If we create a MR db for the anonymous t^1 and t^2, it will refer to the 2-anonymous MR^{root} in Lemma 3.) Next, in line 4 of the *anonymizeSets* algorithm, the anonymization function is called on each pair of their connections in T_1^1 and T_1^2. (Databases returned from these calls correspond to 2-anonymous MR^a databases of Lemma 3.) Returned anonymous dbs are first merged in line 7 of *anonymizeSets* and then concatenated with the anonymous tuples in line 11 as in Lemma 3. ($MR^* = MR^{root}||(\bigcup_i MR^i)$) Since operations are propagated through those tuples of T_1^1 and T_1^2 joined with t^1 and t^2, equivalence classes are explicitly matched through the connected tables. The final output is 2-anonymous by Lemma 3. $\square$

Theorem 5: MiRaCle, when given an input database MR and appropriate parameters, produces a k-anonymous database MR^*.

PROOF. The skeleton of MiRaCle is a clustering-based k-anonymity algorithm. The only change MiRaCle introduces is to call $Anonymize(\sigma_{vip^1=v_1}PT^1, \sigma_{vip^2=v_2}PT^2, MR^1, MR^2)$ lines 9 and 15 for the anonymization of two private trees rooted at v_1 and v_2. Here each private tree is actually a cluster representative for multiple trees. Nodes in each representative tree may have values from higher domains in the given dgh structure (values such as "Science", "Social"). However, such difference does not have any effect on the execution of the anonymize function since the generalization function gen is well-defined also on higher domains (gen("Science","Math")="Science"). The MR^* database returned by the anonymization function will still be anonymous with respect to both trees. Specifically if $v_1 \in MR^1$ and $v_2 \in MR^2$ are m and n anonymous vip representations respectively then $v_3 \in MR^*$=$anonymize(v_1, v_2, MR^1, MR^2)$ is an $m+n$ anonymous representation. At the end of the MiRaCle algorithm, every cluster C has more than k elements and the associated cluster representative v_C is a $|C|$-anonymous

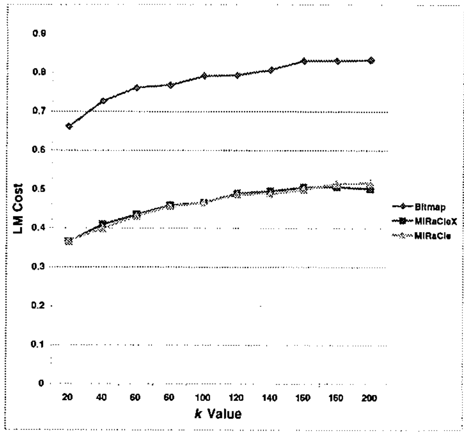

Fig. 4. LM cost for varying k

figure

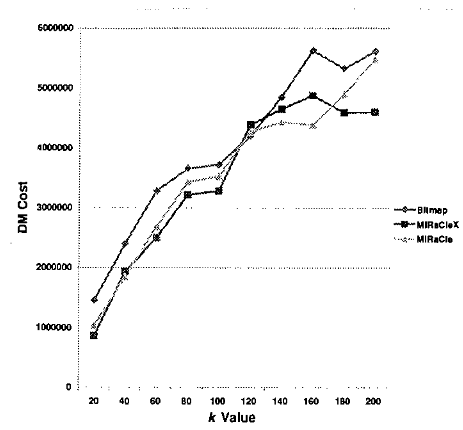

Fig. 6. DM cost for incomplete data

figure

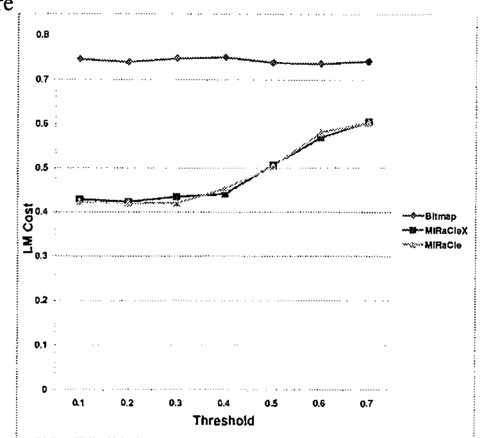

Fig. 5. LM Cost for varying threshold

figure

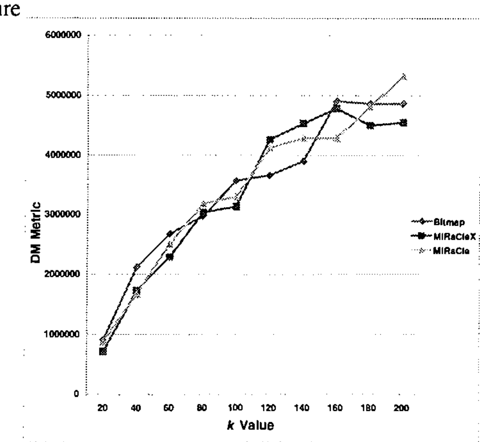

Fig. 7. DM cost for complete data

figure

representative. v_C for each C is reproduced for every entity within C (so that they form an equivalence class). This ensures k-anonymity. So Theorem 4 also implies the correctness of Theorem 5. □

V. Experiments

To compare the flexibility of MiRaCle, MiRaCleX and single-table (bitmap) approach, we conducted experiments on synthetic data structured as in Table II. We created 1000 random students; to each student we assigned 1 obligatory, 2 or 3 technical elective, and 2 or 3 non-technical electives from 22 courses. Each course had 2, 3 or 4 textbooks to choose from. The distribution of courses and books to students was designed to match Bilkent University's undergraduate program requirements. We ran MiRaCle and MiRaCleX on the original database and the CDGH anonymization algorithm [15] on a bitmap transformation of the database. We fixed the cluster limit to be 150. To evaluate the utility of the anonymizations, we used the adaptations of the LM and DM cost metrics defined in Section II.

To observe how MiRaCle and MiRaCleX algorithms address weaknesses given in items 2 and 5 of Section III-C, we first assumed that the dataset is incomplete as described in Section III-C. In Figure 4, we graph the change in LM costs of three anonymizations with respect to different k. Both MiRaCle and MiRaCleX are 30-40% less costly than the Bitmap algorithm. Figure 5 supports the same relation for a fixed $k = 50$ but with varying threshold (clustering input parameter). Figure 6 shows the DM costs for the algorithms. MiRaCle and MiRaCleX slightly outperform the Bitmap algorithm on the DM metric.

We next conducted experiments assuming that the dataset is complete. LM is not a suitable metric for comparison here since it does not take into account tuples that are not in the dataset. Figure 7 shows the DM cost results. We see that all three algorithms have similar costs and there is no obvious winner. The MiRaCle algorithm loses its flexibility advantage discussed in item 3 of Section III-C. This is due

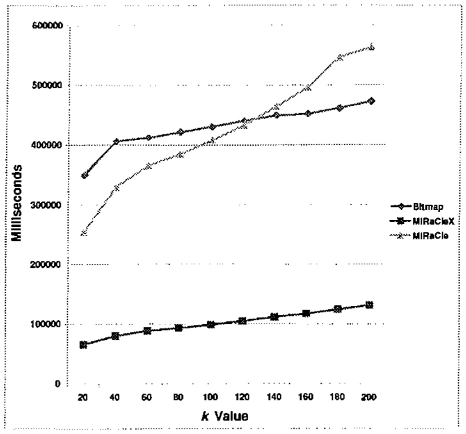

Fig. 8. Execution Time

figure

to the fact that entity anonymizations of MiRaCle are not optimal which means there are cases where Bitmap approach is better w.r.t. precision. However, in Figure 8, we plot the execution time required to run both algorithms on a 1.66GHz Intel Core Duo machine. Consistent with the disscusion in items 6 and 8 of Section III-C, MiRaCleX outperforms both algorithms by a factor of at least 3 (This is true even though we ignored the time spent to convert the dataset to the bitmap format for bitmap anonymizations.) It should be noted that execution times in all conducted experiments show similar behavior. One important observation here is that MiRaCleX have better or comparable utilization when compared to MiRaCle and Bitmap algorithms in all of the experiments however MiRaCleX is much faster than both algorithms. This implies that underlying heuristic works for the experimental dataset.

VI. Conclusions

We have shown that in a full database setting, single table k-anonymity algorithms either fail to protect privacy, or overly reduce the utility of the data. We proposed a more flexible anonymity algorithm for snowflake schemas. Support for arbitrary schemas with multiple private entities can be considered as future work. Other proposed extensions to k-anonymity such as weak k-anonymity [2], ℓ-diversity [13], t-closeness [12], δ-presence [14] application specific k-anonymity [3], distributed k-anonymity [22], and personalized anonymity [20] face similar challenges when considering multi-relational k-anonymity.

References

[1] G. Agrawal, T. Feder, K. Kenthapadi, S. Khuller, R. Panigrahy, D. Thomas, and A. Zhu, "Achieving anonymity via clustering," in *PODS '06: Proc. of the 25th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, Chicago, IL, USA, June 26-28 2006, pp. 153–162.

[2] M. Atzori, "Weak -anonymity: A low-distortion model for protecting privacy." in *ISC*, 2006, pp. 60–71.

[3] K. W. B. Fung and P. Yu, "Top-down specialization for information and privacy preservation," in *Proc. of the 21st Int'l Conf. on Data Engineering*, 2005.

[4] R. Bayardo and R. Agrawal, "Data privacy through optimal k-anonymization," in *Proc. of the 21st Int'l Conf. on Data Engineering*, 2005.

[5] J. Domingo-Ferrer and V. Torra, "Ordinal, continuous and heterogeneous k-anonymity through microaggregation," *Data Min. Knowl. Discov.*, vol. 11, no. 2, pp. 195–212, 2005.

[6] A. O. hrn and L. Ohno-Machado, "Using boolean reasoning to anonymize databases," *Artificial Intelligence in Medicine*, vol. 15, no. 3, pp. 235–254, Mar. 1999. http://dx.doi.org/10.1016/S0933-3657(98)00056-6

[7] A. Hundepool and L. Willenborg, "μ and t-argus: software for statistical disclosure control," in *Third International Seminar on Statistical Confidentiality*, 1996.

[8] V. Iyengar, "Transforming data to satisfy privacy constraints," in *Proc., the Eigth ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining*, 2002, pp. 279–288.

[9] D. Kifer and J. Gehrke, "Injecting utility into anonymized datasets," in *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM Press, 2006, pp. 217–228.

[10] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," in *Proc. of the 2005 ACM SIGMOD Int'l Conf. on Management of Data*, Baltimore, MD, June 13-16 2005.

[11] K. LeFevre, D. DeWitt, and R. Ramakrishnan, "Multidimensional k-anonymity," University of Wisconsin, Madison, Tech. Rep. 1521, June 2005. http://www.cs.wisc.edu/techreports/2005/TR1521.pdf

[12] N. Li and T. Li, "t-closeness: Privacy beyond k-anonymity and l-diversity," in *Proceedings of the 23nd International Conference on Data Engineering (ICDE '07)*, Istanbul, Turkey, Apr. 16-20 2007.

[13] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam, "l-diversity: Privacy beyond k-anonymity," in *Proc. of the 22nd IEEE Int'l Conf. on Data Engineering (ICDE 2006)*, Atlanta Georgia, Apr. 2006.

[14] M. E. Nergiz, M. Atzori, and C. Clifton, "Hiding the presence of individuals in shared databases," in *SIGMOD '07: Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, Beijing, China, June 11-14 2007.

[15] M. E. Nergiz and C. Clifton, "Thoughts on k-anonymization," in *ICDEW '06: Proc. of the 22nd Int'l Conf. on Data Engineering Workshops*. Washington, DC, USA: IEEE Computer Society, 2006, p. 96.

[16] P. Samarati, "Protecting respondents' identities in microdata release," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, 2001.

[17] L. Sweeney, "Guaranteeing anonymity when sharing medical data, the datafly system," in *Proc., Journal of the American Medical Informatics Association*. Hanley & Belfus, Inc., 1997.

[18] L. Sweeney, "Achieving k-anonymity privacy protection using generalization and suppression," *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, no. 5, 2002.

[19] L. Sweeney, "k-anonymity: a model for protecting privacy," *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, vol. 10, no. 5, pp. 557–570, 2002.

[20] X. Xiao and Y. Tao, "Personalized privacy preservation," in *SIGMOD '06: Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of data*. New York, NY, USA: ACM Press, 2006, pp. 229–240.

[21] C. Yao, X. S. Wang, and S. Jajodia, "Checking for k-anonymity violation by views," in *VLDB '05: Proc. of the 31st Int'l Conf. on Very large data bases*. VLDB Endowment, 2005, pp. 910–921.

[22] S. Zhong, Z. Yang, and R. N. Wright, "Privacy-enhancing k-anonymization of customer data," in *PODS '05: Proc. of the 24th ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. New York, NY, USA: ACM Press, 2005, pp. 139–147.