Checking for k-Anonymity Violation by Views*

Chao Yao
Ctr. for Secure Info. Sys.
George Mason University
cyao@gmu.edu

X. Sean Wang
Dept. of Computer Sci.
University of Vermont
Sean.Wang@uvm.edu

Sushil Jajodia
Ctr. for Secure Info. Sys.
George Mason University
jajodia@gmu.edu

Abstract

When a private relational table is published using views, secrecy or privacy may be violated. This paper uses a formally-defined notion of k-anonymity to measure disclosure by views, where $k>1$ is a positive integer. Intuitively, violation of k-anonymity occurs when a particular attribute value of an entity can be determined to be among less than k possibilities by using the views together with the schema information of the private table. The paper shows that, in general, whether a set of views violates k-anonymity is a computationally hard problem. Subcases are identified and their computational complexities discussed. Especially interesting are those subcases that yield polynomial checking algorithms (in the number of tuples in the views). The paper also provides an efficient conservative algorithm that checks for necessary conditions for k-anonymity violation.

1 Introduction

In this paper, we are concerned with data release (or data publication) from relational databases. We assume that secret and private information takes the form of associations, that is, pairs of values appearing in the same tuple. For example, the association of "John" with "Obesity" in a medical database is private information. Note that neither "John" nor "Obesity" alone is a secret, but the association of the two values is. Instead of releasing the whole relational table that contains secret associations, we publish query results on the table.

The released data can thus be described as materialized views. Secret associations themselves can also be defined as a view with the projection on the two attributes involved in a secret or private associations. For example, in Figure 1, given base table P_1, secret associations can be defined by $\Pi_{Name,Problem}(P_1)$, while released data are two views, namely, $\Pi_{Name,Job}(P_1)$ and $\Pi_{Job,Problem}(P_1)$.

Name	Job	Salary	Problem
George	Manager	70K	Cold
John	Manager	90K	Obesity
Bill	Lawyer	11K	HIV

Base table P_1

$v_1 = \Pi_{Name,Job}(P_1)$

Name	Job
George	Manager
John	Manager
Bill	Lawyer

$v_2 = \Pi_{Job,Problem}(P_1)$

Job	Problem
Manager	Cold
Manager	Obesity
Lawyer	HIV

Figure 1: Base table and two releasing views.

Note views here are used for releasing data instead of for enforcing access control policies (as in, e.g., [10]).

A simple protection we may immediately come up with is that we do not allow access to the two attributes in secret associations together in one view. However, in many cases, releasing the two attributes in two different views may still cause problems. Indeed, for the example in Figure 1, it is not difficult to deduce that "Bill" has "HIV" from the two released views v_1 and v_2, since "Bill" can only be linked to "HIV" by the value "Lawyer" on the common attribute between v_1 and v_2. The purpose of this paper is to study how to protect secret associations from leaking through multiple views.

Data release in most cases requires tolerance of some disclosure (see [8]). A practical approach is to provide a measure of disclosure and, based on this measure, organizations can give a threshold on how much disclosure is tolerated.

*The work was partially supported by the NSF grants IIS-0430402, IIS-0430165, and IIS-0242237.

Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005

In short, this paper addresses two problems: (1) how to define a measure of information disclosure by a set of releasing views regarding secret associations, and (2) how to determine whether a set of releasing views discloses more information about secret associations than a predefined threshold in terms of the defined measure.

There already exists work addressing the problems similar to the above ones. Based on a probability model, Miklau and Suciu suggested a measure on information disclosure by a set of views w.r.t. a given secret view [8]. But this measure is to calculate the total information disclosure about a secret view. It can happen that the disclosure about a single association is severe, but the sum of the disclosure for all secret associations is relatively minor. In practice, disclosure about a single association matters even if the total disclosure is tolerable. Furthermore, in most cases, the calculation of disclosure based on this measure is far too hard to be practical; and reasonable and feasible necessary conditions for the disclosure above the threshold by this measure do not seem to exist.

In this paper, we apply a different measure, namely k-anonymity, on the disclosure about each single secret association; and we provide a set of practical checking methods. The concept of k-anonymity was introduced in [12] to anonymize a single releasing view. We give a formal definition of k-anonymity based on relational view. And then we concentrate on how to detect whether or not a given set of releasing views violates k-anonymity. We study the cases where checking is polynomial time and the cases where checking is intractable. For the former, checking methods are presented; for the latter, we suggest an efficient conservative checking method based on "indistinguishability" between tuples in the releasing views.

The rest of paper is organized as follows. In the next section we provide some intuition about k-anonymity and related issues. We give our formal definitions and notation in Section 3. The basic mechanism of our checking methods is presented in Section 4. We provide the checking methods for the cases without and with functional dependencies in Section 5 and Section 6, respectively. The conservative checking method is introduced in Section 7. We discuss the related work in Section 8, and conclude with Section 9.

2 Discussion

In this section we discuss the notion of k-anonymity in an intuitive manner and provide some reasons why it is nontrivial to check releasing views for k-anonymity violation. Roughly speaking, k-anonymity means that one can only be certain that a value is associated with one of at least k values. Such collections of values give anonymity to secret associations. For example, from the releasing views in Figure 1, outside users can know that George has at least one of two problems "Cold" and "Obesity", but outside users do not know which one is certainly his. We can say the releasing views do not violate 2-anonymity regarding George's privacy. If the views do not violate k-anonymity for a person, where k is predefined per privacy policy, then the disclosure of that person's privacy is considered tolerable.

$\Pi_{Name}\sigma_{Salary>80K}(P_1)$

Name
John
Bill

$\Pi_{Problem}\sigma_{80K<Salary<100K}(P_1)$

Problem
Obesity

Name
George
John

$\Pi_{Name}\sigma_{Salary<105K}(P_1)$

Figure 2: Three releasing views on P_1. Selection conditions play a role in disclosure.

Checking releasing views for k-anonymity turns out nontrivial in general. Firstly, releasing views may not merely be projections but with selection conditions. Such selection conditions make checking difficult. As an example, in Figure 2, there are three releasing views on table P_1 in Figure 1. If the first view is intersected with the second one, the third view satisfies the selection condition of the intersection. Thus, "John" is revealed to be associated with "Obesity", violating 2-anonymity. Generally, the operations used for inference can be join, intersection, union, and difference, making the situation complicated.

Name	Problem	Charge
George	Cold	20K
John	Obesity	20K
John	Obesity	30K
Bill	HIV	30K

Base table P_2

$\Pi_{Name,Charge}(P_2)$

Name	Charge
George	20K
John	20K
John	30K
Bill	30K

$\Pi_{Charge,Problem}(P_2)$

Charge	Problem
20K	Cold
20K	Obesity
30K	Obesity
30K	HIV

Figure 3: Base table P_2 with an FD $Name \rightarrow Problem$ and 2 releasing views. The FD affects anonymity.

Secondly, FDs (Functional Dependencies) may exist across views, which in some situations cannot be assumed to be unknown to outside users. Such FDs may provide more ways to infer the information about secret associations. For example, in Figure 3, there are two releasing views with an FD $Name \rightarrow Problem$ across them. Since "John" can be associated with only one problem, by the given FD, "John" must be associated with "Obesity". Since only "obesity" appears in both tuples in the second view that can join the "John" tuples in the first view. Thus violation of privacy is made possible due to the FD.

3 Formal Definitions

In this paper, we only consider views on a single private table. We believe this covers many cases in practice. We also restrict the releasing views to those that contain only (duplicate removing) projections and selections. And we do not consider the situation in which the base table may be updated.

We assume a base table *baseTbl* with schema D, which consists of a set of attributes and a (possible empty) set of FDs (Functional Dependencies). We assume no NULL values are allowed, and will use $\mathcal{I}^{\mathcal{D}}$ to denote the set of all possible relations on D. By this definition, FDs are the only possible constraints on *baseTbl*. We further assume that D contains two attributes *ID* and P, where a value on *ID* is an identifier of an external entity (e.g., a person) while a value on P is a sensitive property of the entity. Although, *baseTbl* may contain many attributes that can be viewed as *ID* and P, in the sequel we assume a pair of fixed *ID* and P, and we call $S = \Pi_{ID,P}(baseTbl)$ the *secret view* on *baseTbl*.

We consider the situation where data in *baseTbl* are being released in the form of a view set. A *view set* is pair (V, v), where V is a list of selection-projection queries $(q_1, q_2, ..., q_n)$ on *baseTbl*, and v is a list of relations $(r_1, r_2, ..., r_n)$ without duplicate tuples such that $\exists I \in \mathcal{I}^D$, $r_i = q_i(I)$ for each $i = 1, 2, ..., n$. We may abbreviate (V, v) to v if V is understood. We call a pair (q_i, r_i) a *view* in the view set (V, v), if q_i is in V and r_i is the corresponding relation in v. We also denote (q_i, r_i) by v_i. We suppose there exist views (q_i, r_i) and (q_j, r_j) in v such that q_i and q_j contain the projection on *ID* and P, respectively. In order to avoid confusion with the tuples in *baseTbl*, a tuple in a view is called a PF (Projection Fact), the same term used in [2].

Other than the view set, outside users only know the base table schema D with its functional dependencies if any. Given a view set v, outside users can deduce the possible base table instances that can yield v. The set of all these instances is denoted as $\mathcal{I}^v$, i.e. $\mathcal{I}^v = \{I \in \mathcal{I}^{\mathcal{D}} | \forall (q_i, r_i) \in v : r_i = q_i(I)\}$. We use I to denote an instance in $\mathcal{I}^v$, and $S(I)$ to denote the secret view on I, i.e., $S(I) = \Pi_{ID,P}(I)$. We will use T^v to denote the union of all instances in $\mathcal{I}^v$, i.e., the set all of possible tuples in all instances in $\mathcal{I}^v$.

Binary tuples in $\Pi_{ID,P}(baseTbl)$, also called *secret associations*, are what need to be protected. In Figure 1, *baseTbl* is P_1, v consists of two views v_1 and v_2, and S corresponds to $\Pi_{Name,Problem}(P_1)$. Since *(John, Obesity)* is in the secret view from P_1, we would like to protect from outside users the fact that *John* is associated with *Obesity*. "Protect" is a notion defined precisely below via "anonymity".

Definition (**association cover**) Each binary tuple on (ID, P) is called an *association*. Given a view set v, a set A of associations on (ID, P) is called an *association cover w.r.t. v* if all the binary tuples in A have the same *ID* value and for each I in $\mathcal{I}^v$, $S(I) \cap A \neq \emptyset$.

Each association cover has the same value on *ID*. An association cover of size k is called a *k-association-cover*. An association cover is *minimal* if none of its proper subsets are association covers. In Figure 1, {*(John, Cold), (John, Obesity)*} is an association cover. Indeed, *John* is a manager from v_1 while a manager has either *Cold* or *Obesity*, or both from v_2. Hence, for any base table instance I that yields v_1 and v_2, either *(John, Cold)* or *(John, Obesity)*, or both are in $S(I)$. By definition, {*(John, Cold),(John, Obesity)*} is 2-association-cover. This cover is minimal since neither {*(John, Cold)*} nor {*(John, Obesity)*} is an association cover.

Definition (**k-anonymity**) Given a view set v and integer $k \geq 2$, we say v violates *k-anonymity* if there exists an association cover w.r.t. v of size less than k.

Intuitively, if a view set does not violate k-anonymity (for a user-specified, sufficiently large k), then we would say that all the secret associations are "protected". This definition requires that for each value a on *ID*, there does not exist an association cover with *ID* being a; in other words, it requires k-anonymity *for* each a on *ID*.

By definition, if a view set violates k-anonymity, then there exists an n-association-cover such that $n < k$. An extreme case is when 2-anonymity is violated, and in this case we surely know a binary tuple on (ID, P), the one in the association cover, is a secret association, i.e., it must be in the secret view on *baseTbl* (actually on any allowable instance that yields v). In Figure 1, the view set of v_1 and v_2 violates 2-anonymity, since {*(Bill, HIV)*} is 1-association-cover. This means *(Bill, HIV)* must be a secret association.

Proposition 1 *Given a view set v, if A is a minimal association cover, then for each association α in A, there exists I in $\mathcal{I}^v$ such that $\alpha \in S(I)$ and $(A - \{\alpha\}) \cap S(I) = \emptyset$.*[1]

Proposition 1 says that for each given association in a minimal association cover, there exists an instance (that yields v) that contains the given association but no other associations in the association cover. This provides an intuition for k-anonymity. By definition, if a view set does not violate k-anonymity, any of its association covers must have at least k associations in it. It means that outside users can never make sure which one of k property values an entity e is certainly associated with, except when they are be able to exclude $k - 1$ values from them using some external knowledge. Indeed, even if they can exclude $k - 2$ values, by

[1] Proofs are omitted due to space limit, but can be found at http://mason.gmu.edu/∼cyao/kTechnical.pdf

proposition 1, there still exist two possible instances such that each contains exactly one of the two remaining associations; thus, the outside users still cannot make sure which one of the values must be associated with entity e. Therefore, each entity's secrecy is protected by a kind of anonymity. The greater the k is, the more protection the secret has.

Notation	Explanation
$baseTbl$	Current Base Table
D	Base Table Schema
$\mathcal{I}^D$	All allowable base table instances on D
(v, V) or v	View set
$\mathcal{I}^v$	All allowable base table instances yielding v
T^v	Union of all instances in $\mathcal{I}^v$
$S = \Pi_{ID,P}(baseTbl)$	Secret view definition
α	Association
A	Association Cover

Table 1: Notation table.

4 Basic Mechanism

In this section we turn to methods that check whether a view set violates k-anonymity. By definition, it is equivalent to checking whether or not there are association covers of size less than k.

If there are no FDs in D, we will show that the time complexity of checking is polynomial. Otherwise, checking generally is very intractable. In fact, we will show that its complexity is Σ_2^p-hard. Note the complexity here is "data complexity", i.e., we only consider the size of the relations in the view set as input size. In this paper, we will consider special cases where checking can be done in polynomial time, or can be approximated. (As noted in [13], Σ_2^p-hard problems are not easily approximated. Hence it is not easy to find a good approximation for the general case.)

In Section 5, we first give the checking algorithm for the case where FDs are not present. This algorithm also forms the basis for checking methods in Section 6 for the case where FDs are present. We then consider the subcases where there are no selection conditions in the queries in the view sets, which is called the *projection-only subcase*. We believe this kind of view set is used frequently in practice.

A view set can be looked as constraints on the base table, since each instance must exactly return the PFs (Projection Facts) in the view set. More specifically, give a view set v, by the definition of $\mathcal{I}^v$, we have (1) For each PF p in the view (q_i, r_i) and each instance I in $\mathcal{I}^v$, a tuple t must exist in I such that $p = q_i(t)$; (2) For each tuple t' and each view (q_i, r_i), if $p' = q_i(t')$ and p' is not in r_i, then t' is not in any instance in $\mathcal{I}^v$.

For example, consider the base table P_1 in Figure 1, and the view $(\Pi_{Name}\sigma_{Salary>80K}(baseTbl), \{(John); (Bill)\})$. By property (1) above, for each instance I in $\mathcal{I}^v$, one tuple of the form $\{(John, j, s, p)\}$, where s is a value greater than 80k and j and p are any allowable values, must be in I. Furthermore, by property (2), a tuple like $(George, manager, 90K, Cold)$ cannot be in T^v since otherwise $George$ would have to appear in the view.

To precisely describe the above constraints given by a view set, we introduce the following concepts.

Definition (**Tuple Cover**) Given a view set v, a *tuple cover T for v* is a set of tuples on D such that for each I in $\mathcal{I}^v$, $I \cap T \neq \emptyset$.

Tuple covers have a similar meaning as association covers. Note $\Pi_{ID,P}(T)$ is an association cover for each tuple cover T, since for each I in $\mathcal{I}^v$, if $I \cap T \neq \emptyset$, then $\Pi_{ID,P}(I) \cap \Pi_{ID,P}(T) \neq \emptyset$. Thus, *we can get all minimal association covers by obtaining all minimal tuple covers.*

Definition (**Tuple Set for a PF**) Given a PF p in a view (q_i, r_i) in v, *the tuple set for p* is the set of all the tuples t in T^v such that $q_i(t) = p$.

The tuple set for a PF is all the tuples that exist in any possible instance and yield the given PF with the corresponding query. It is easily seen that the tuple set for a PF is a tuple cover. Indeed, given an instance I in $\mathcal{I}^v$, there must be a tuple t in I such that $q_i(t) = p$ by the definition of $\mathcal{I}^v$, and therefore the tuple set for p intersects with I. Therefore, *to get all minimal tuple covers, we may first get all tuple sets for all PFs in v.*

Before presenting the basic checking method, we introduce some more notation for convenience. Given a view set v, we define the following sets.

- $\mathcal{A}_{min}$: the set of all minimal association covers;
- $\mathcal{T}_{min}$: the set of all minimal tuple covers;
- $u(p)$: the tuple set for PF p in v;
- U: the set of all tuple sets $u(p)$ for all PFs in v;
- $w(p)$: the tuple set for PF p in v disregarding FDs. Here "disregarding FDs" means that we assume that there were no FDs on $baseTble$ even if there are. That is, as far as $w(p)$ and W below are concerned, we consider $\mathcal{I}^v$ to consist of all possible instances on $baseTbl$ (that yield v) that do or do not satisfy the FDs. In case we do not have FDs on $baseTbl$ to start with, $u(p) = w(p)$.
- W: the set of all tuple sets $w(p)$ for all PFS in v disregarding FDs.

Proposition 2 *Given a view set v, (a) $\mathcal{A}_{min} \subseteq \{\Pi_{ID,P}(T) | T \in \mathcal{T}_{min}\}$, (b) $u(p) \subseteq w(p)$ for each p in v.*

Proposition 2(a) says that the set of all minimal association covers are a subset of the set consisting of all the projections of all minimal tuple covers upon ID and P. Proposition 2(b) says that for each PF p in v,

the tuple set for p is a subset of that disregarding all the FDs.

Our main checking methods are stimulated by the above proposition. The outline of the methods are illustrated by the following diagram.

$$W \Rightarrow U \Rightarrow \mathcal{T}_{min} \Rightarrow \mathcal{A}_{min}$$

That is, we consider the view set without FDs to compute W, explained in Section 5. Such a computing method is also a checking method for the case without FDs. Then we present the methods inspired by proposition 2(b) that impose FDs on W to get U in Section 6 for some subcases. Also, we introduce some methods that compute $\mathcal{T}_{min}$ from U in Section 6. Finally we can get $\mathcal{A}_{min}$ from $\mathcal{T}_{min}$ by Proposition 2(a).

5 Without FDs

This section details a checking method for the case where there are no FDs on *baseTbl*. This checking method in effect gives W from a given view set v. The time complexity of the method is polynomial in the number of the tuples in v (the number views and the size of the view definitions are take as constants).

5.1 Checking method

Since there are no FDs on *baseTbl*, we have $U = W$.

Proposition 3 *Given a view set v, for two distinct PFs p_1 and p_2 in the same view in v, $w(p_1)$ and $w(p_2)$ are disjoint.*

Since p_1 and p_2 are distinct tuples in the same view, there must exists an attribute $\hat{A}$ such that the values of p_1 and p_2 on $\hat{A}$ are distinct. Then $w(p_1)$ and $w(p_2)$ must be disjoint.

Proposition 4 *For a view set v on baseTbl without FDs, $\mathcal{T}_{min} \subseteq W$.*

The above proposition forms the basis of our checking method for this case. By Proposition 2(a), $\{\Pi_{ID,P}(\mathcal{T}) | \mathcal{T} \in \mathcal{T}_{min}\}$ is a superset of $\mathcal{A}_{min}$. Since $\mathcal{T}_{min}$ is a subset of W, $\{\Pi_{ID,P}(w(p)) | w(p) \in W\}$ is a super set of $\mathcal{A}_{min}$. Therefore, the problem of finding all the minimal association covers becomes that of finding W, all tuple sets for all the PFs.

Now we show a procedure for finding W. Firstly, for each (q_i, r_i) in v, let v^i be the view set containing the single view (q_i, r_i). Given v^i, we generate tuple set T_{ij} for each p_{ij} in r_i and a *complement tuple set* T_{i0} as follows, where $j = 1, 2, ...$, (1) $T_{ij} = \{t | q_i(t) = p_{ij}, t \in T^{v^i}\}$; (2) T_{i0} is the set of all allowable tuples appearing in $\mathcal{I}^{v^i}$ that make the selection condition of q_i false. These tuple sets are called *tuple sets by single view*. We will use logical expressions to represent these tuple sets. We use $F_{ij} = C_i \wedge p_{ij}$ to express T_{ij}, where C_i is the selection condition of q_i, and p_{ij} is the conjunction $\wedge_x(\hat{A}_x = \Pi_{\hat{A}_x}(p_{ij}))$, where x ranges over all attributes $\hat{A}_x$ in p_{ij}. Clearly, a tuple t is in T_{ij} iff the attribute values of t satisfy the condition $C_i \wedge p_{ij}$, since there are no FDs. Similarly, we use $F_{i0} = \neg C_i$, the complement to the selection condition of q_i, to express T_{i0}.

For example, consider the view $(\Pi_{Name}\sigma_{Salary>80K}(baseTbl), \{(John); (Bill)\})$ in Figure 2, which is also shown in in Figure 4. (Here we omit out the attribute *Job* in *baseTble* for convenience, since it does not affect the checking result.) We generate two tuple sets by single view for the PF $(John)$ and $(Bill)$, respectively, and the dump tuple set. The tuple set by single view for $John$ is represented by $(salary > 80K) \wedge (name = John)$. In our example, in order to make easier illustration, we shall use another intuitive notation $(John, s > 80K, *)$ to denote the same tuple set. Here $(John, s > 80K, *)$ denotes the set of tuples that the first entry value is $John$, the second is a number greater than $80K$ and the third is any allowable value. Similarly, the tuple set for PF $Bill$ can be written as $(Bill, s > 80K, *)$; and the complement tuple set is denoted by $(*, s \leq 80K, *)$.

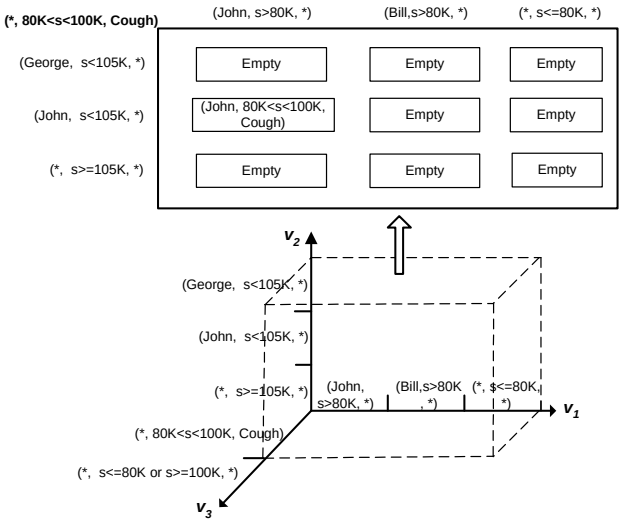
For each view v_i in v, all tuple sets by v_i including the complement tuple set are pairwise disjoint by Proposition 3. Let $Z_i = (\cup_{T_{ij} \in T^i} T_{ij}) \cup T_{i0}$, where T^i is the set of all tuple sets by single view for all PFs in v_i. We can now generate $w(p_{ij})$ via the following proposition.

Proposition 5 *Given a view set v, $w(p_{ij}) = (T_{ij} \cap (\cap_{m \neq i} Z_m))$, for each PF p_{ij} in (q_i, r_i) in v.*

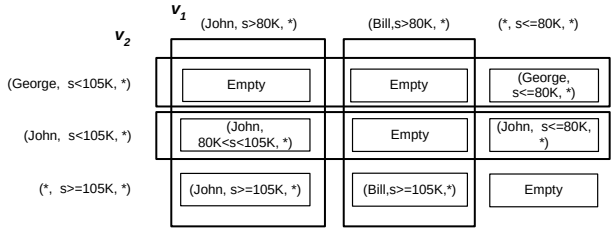
Proposition 5 says that for each PF p in one view in v, to get the tuple sets for p, we can intersect the tuple set by single view for p with the unions of all the tuple sets by single view for the PFs in the other views. For example, given the 3-view set in Figure 2, let T_{ij} denote a tuple set only by v_i and T_{i0} denote a complement tuple set, where $j > 0$ and $i = 1, 2, 3$. Then the tuple set for the PF $(John)$ in v_1 is $T_{11} \cap (T_{21} \cup T_{22} \cup T_{20}) \cap (T_{31} \cup T_{32} \cup T_{30})$.

On considering the generation of all tuple sets for PFs, we can improve this method by merging the common calculations between the tuple sets. For example, if the tuple set for p_{11} is formed by $T_{11} \cap (T_{21} \cup T_{20}) \cap (T_{31} \cup T_{30})$, it is calculated by the equivalent formula $(T_{11} \cap T_{21} \cap T_{31}) \cup (T_{11} \cap T_{21} \cap T_{30}) \cup (T_{11} \cap T_{20} \cap T_{31}) \cup (T_{11} \cap T_{20} \cap T_{30})$. In other words, each views in a given view set can be looked as one dimension, in which each tuple set is a coordinate, since they are disjoint by Proposition 3. Each point or cell, corresponding to a clause in the above example, is formed by intersecting the corresponding coordinates. Then the tuple sets for PFs can be obtained by projecting corresponding cells. Thus, the common calculations that generate the cells need not be repeated.

By Proposition 2 and Proposition 4, after obtaining the tuple sets $w(p_{ij})$ for all PFs in v, we project



(a) views v_1, v_2 and v_3



(b) views v_1 and v_2

Figure 4: The illustration of checking procedure for the view set in Figure 2

them upon *ID* and P to obtain all minimal association covers. Finally, we check these minimal association covers to determine whether the view set violates k-anonymity or not by looking at the sizes of the association covers. This is equivalent to checking whether or not $|\Pi_{ID}(w(p_{ij}))| = 1$ and $|\Pi_P(w(p_{ij}))| < k$ for each PF p_{ij}. Here we make another improvement on the method. That is, we make such projections upon (ID, P) performed in advance. Since the tuple sets are the union of corresponding cells, we can project each cell upon *ID* and P, respectively, and then union the projections. Moreover, since we only need to know whether or not $|\Pi_{ID}(w(p_{ij}))| = 1$ and $|\Pi_P(w(p_{ij}))| < k$, if for a given cell, the number of distinct projections on *ID* is greater than 1, or the number of distinct projections on P is greater than or equal to k, or they are just empty, we may just mark the cell with a special label and do not need to enumerate the values. Indeed, if one of the cells that form $w(p_{ij})$ is specially labelled, $|\Pi_{ID}(w(p_{ij}))| = 1 \wedge |\Pi_P(w(p_{ij}))| < k$ is not true. Now we concentrate on how to project and enumerate the values on *ID* and P for each cell. Since each cell is formed by the conjunction of logical expressions that represents the tuple sets by single view, the projections are not conventional relation projections, but is equivalent to quantifier elimination [9]. The result of quantifier elimination here is a formula having a single variable that represents *ID* or P. Thus, we can determine how many values on *ID* or P can satisfy this kind of formula, and enumerate them. The complete description of the algorithm is in Figure 5. In the figure, given i and j, the expression $(\cup_{m_1,...,m_{i-1},m_{i+1},...,m_N} (c^{ID}_{m_1...m_{i-1}jm_{i+1}..m_N}))$ means the union of all sets denoted by $c^{ID}_{...j...}$ that have the subscript of the j value at the ith position. By the above study, we can see that the following theorem holds.

Theorem 1 *Given a view set v and integer k, the procedure in Figure 5 correctly checks v for the violation of k-anonymity.*

Procedure *Checking k-anonymity*
Input: v, $S = \Pi_{ID,P}$ and integer k
Output: True (violating k-anonymity) or False
Let N be the number of views in v.
Let n_i be the number of PFs in view v_i in v
For each view v_i with selection condition C_i
Let $F_{i0} = (\neg C_i)$
For each PF p_j in the view v_i
Let $F_{ij} = (C_i \wedge p_j)$
For each m_1, ..., m_N , where
$0 \leq m_1 \leq n_1$, ..., $0 \leq m_N \leq n_N$
Let $c^{ID}_{m_1...m_N} = \Pi_{ID}(F_{1m_1} \wedge ... \wedge F_{Nm_N})$
Let $c^{P}_{m_1...m_N} = \Pi_{P}(F_{1m_1} \wedge ... \wedge F_{Nm_N})$
For each i,j, where $1 \leq i \leq N$ and $1 \leq j \leq n_i$
Let $w^{ID}_{ij} = \cup_{m_1,...,m_{i-1},m_{i+1},...,m_N}($
$c^{ID}_{m_1...m_{i-1}jm_{i+1}..m_N})$
Let $w^{P}_{ij} = \cup_{m_1,...,m_{i-1},m_{i+1},...,m_N}($
$c^{P}_{m_1...m_{i-1}jm_{i+1}..m_N})$
If $|w^{ID}_{ij}| = 1$ and $|w^{P}_{ij}| < k$
Return true
Return false

Figure 5: A Procedure for checking a view set without FDs for k-anonymity

Figure 4 shows an example of the generation procedure for tuple sets. Given the 3-view set in Figure 2, we generate three sets of the tuple sets by single view, respectively, plus the complement tuple sets, shown as the coordinates of the three dimensional in Figure 4(a), and then intersect them to generate the tuple sets for the PFs. In the top portion of Figure 4(a), we show a tuple set for the PF *(Cough)* in v_3, which is $(John, 80K < s < 100K, Cough)$. In order to illustrate the intersection clearly, Figure 4(b) also shows the intersection of two sets of the tuple sets by v_1 and v_2. We get the tuple sets constrained by the view set $\{v_1, v_2\}$ via the intersection, which are the box with bold lines in the figure.

5.2 Time Complexity and Improvements

Here we are only interested in data complexity with respect to the number of PFs in a given view set v. The number of the views and the sizes of the query expressions in v are taken as constants.

In the algorithm, every operation can be easily implemented except for $c^{ID}_{m_1 \ldots m_N} = \Pi_{ID}(F_{1m_1} \wedge \ldots \wedge F_{Nm_N})$ and $c^{P}_{m_1 \ldots m_N} = \Pi_{P}(F_{1m_1} \wedge \ldots \wedge F_{Nm_N})$ that are implemented by quantifier elimination and enumeration of the satisfied values for logical expressions having a single variable. Its complexity is dependent only on the size and type of formula, and the domain of variables, which are independent on the number of PFs in v. Thus, the time complexity of each projection and enumeration is constant regarding the number of PFs in v. Note that since the selection conditions of the most queries in releasing views are simple types of short expressions, such as conjunctive expressions with inequalities, those projections and enumerations do not cost much time.

The following two steps dominate the computing time. One is the generation of N dimensional cells c. The other is the generation of T'_{ij}. The former has $\prod_i |v_i|$ conjunction operations. The latter has $N \prod_i |v_i|$ union operations. Since the number of views are taken as constant regarding the size of view set (actually, the complexity is in the number of views), the data complexity of the method is $O((max(|v_i|))^N)$. Hence, we have the following theorem.

Theorem 2 *The time complexity of the algorithm in Figure 5 is polynomial in the number of PFs in a given view set.*

Certainly, this algorithm is just a basic one. It can be improved in many ways. For example, if two view sets v_1 and v_2 have the exclusive selection conditions, their tuple sets can be put into the same dimension. Or if v consists of two collections of views that have exclusive selection conditions to each other, by the following proposition, we can check them separately.

Proposition 6 *Given two view sets v and v', if v and v' do not violate k-anonymity and each view of v has exclusive selection condition to each view of v', then the view set $v \cup v'$ does not violate k-anonymity.*

For the projection-only subcase, the above checking procedure can be simplified greatly. Indeed, it can be simplified into an intuitive and straightforward checking method through relational natural join between views. The complement tuple sets are empty and hence can be discarded, since there are no selection conditions for each view. For the same reason, each PF itself already represents the tuple set. By the procedure in Figure 5, each cell c is corresponding to a PF p in the natural join result between the views, which means that c is composed of all allowable tuples that have the projection value p. Therefore, checking can be done by natural join on the views in a given view set. A pitfall we want to note here is that one may think checking can be done as follows. For each value a on ID, count how many values of P are linked to a in the natural join; if the number is less than k, then the view set violates k-anonymity. This intuition is not correct, which is equivalent to only checking the tuple sets for the values on ID. For example, given two views $(\Pi_{ID}(baseTbl), \{a_1\})$ and $(\Pi_{P}(baseTbl), \{b_1, b_2\})$. we cannot just check the tuple set for a_1, which does not result in the violation of 2-anonymity. However, $\{(a_1, b_1)\}$ is an association cover. In contrast, the procedure in Figure 5 catches all the minimal association covers.

6 With FDs

If there are FDs in the base table schema, which outside users are assumed to have knowledge of, checking for k-anonymity is Σ^p_2-hard, since a special subcase is Σ^p_2-complete, which is shown below. We say an FD $\hat{A}_1 \rightarrow \hat{A}_2$ is ***across*** views v_1 and v_2, if v_1 (v_2, respectively) contains a projection on $\hat{A}_1$ ($\hat{A}_2$, respectively) in it or $\hat{A}_1$ ($\hat{A}_2$, respectively) appears in the selection condition of v_1 (v_2, respectively).

Theorem 3 *Given a two-view set v in the projection-only subcase, where there is a single FD across them, and integer k, the problem of whether there exists an association cover of size less than k is NP-hard.*

Theorem 4 *Given a two-view set v in the projection-only subcase, where there are two distinct FDs across them, the problem of whether a set of associations is an association cover is CO-NP-complete; Given integer k, the problem of whether there exists an association cover of size less than k is Σ^p_2-complete.*

Corollary 1 *Given a view set v on baseTbl with FDs, the problem of whether a set of associations is an association cover is CO-NP-hard; Given integer k, the problem of whether there exists an association cover of size less than k is Σ^p_2-hard.*

However, there are still many subcases where checking is polynomial or can be approximated easily. We discuss some typical ones in the following subsections.

6.1 Ineffective FDs for Secret Associations

Not all FDs increase checking complexity. Some FDs have no effects on a given view set regarding k-anonymity. For example, given a view v and an FD F $\hat{A}_1 \rightarrow \hat{A}_2$, where $\hat{A}_1$ and $\hat{A}_2$ only appear in the same view in v, F does not affect v regarding k-anonymity. Given an FD F and V, for each view set (V, v), if the set of minimal association covers w.r.t v without F is the same as that with F, we say F is *ineffective* for V.

We list some typical conditions of ineffective FDs. Given V and an FD F $\hat{A}_1 \rightarrow \hat{A}_2$, F is ineffective for V, if for each two views v_1 and v_2 in (V, v), one of the following conditions is hold:

- F is not across v_1 and v_2;
- F is across v_1 and v_2; and v_1 and v_2 have the same selection condition; and at least one of v_1 and v_2 contains the projection on $(\hat{A}_1, \hat{A}_2)$;
- F is across v_1 and v_2; and v_1 and v_2 have the same selection condition; and there exist $\hat{A}_1 \rightarrow \hat{A}_3$ and $\hat{A}_3 \rightarrow \hat{A}_2$, where $\hat{A}_3$ is contained in both v_1 and v_2;

For example, the queries V in a given view set are $\Pi_{Name,Job}(baseTbl)$ and $\Pi_{Job,Salary}(baseTbl)$. There are FDs $Name \rightarrow Job$, $Job \rightarrow Salary$ and $Name \rightarrow Salary$ in the schema. By the above 1st and 3rd condition, all these FDs are ineffective for V. Thus, for any view set from V, the checking method for the case without FDs can be applied, which is polynomial time.

6.2 Without Common Attributes and with a Single FD

The common attributes between the views in a given view set play a major role in the violation of k-anonymity. If there are no common attributes, checking may be much easier. We first consider a view set with two projection-only views without common attributes and with a single FD across the views. One view contains ID, denoted as (q_{ID}, r_{ID}) or v_{ID}, and another contains P, denoted as (q_P, r_P) or v_P. We have the following conclusion.

Proposition 7 *Given a projection-only view set $\{(q_{ID}, r_{ID}), (q_P, r_P)\}$, where there is only one FD across them and there are no common attributes between them, for each PF p_i in r_{ID} and each PF p_j in r_P, there exists a tuple t in T^v such that $q_{ID}(t) = p_i$ and $q_P(t) = p_j$.*

This proposition claims that each PF in v_{ID} can be *linked* to each PF in v_P. Here we say a PF can be *linked* to another PF, if there exists a tuple t in T^v such that t yields both PFs. (A checking procedure can be looked as determining the links between the PFs in v.) Thus, if the single FD is $ID \rightarrow P$, then checking is trivial, which just counts how many distinct values on P in r_P. Otherwise, checking is NP-complete due to the FD, as shown in the proof of Theorem 3.

Now we use an example in Figure 6 to illustrate why checking is NP-hard for this subcase. There is a two-view set v in Figure 6; attribute A is ID and attribute D is P. Since the number of distinct values on B is the same as that on C in v, due to the FD $B \rightarrow C$, each value on B has to be associated with one and only one value on C in each instance in $\mathcal{I}^v$. Consider the association covers with a_1. Since the PFs p_{11} and p_{12} containing a_1 have two distinct values on B, they have to be linked to the PFs in the second view with exact two distinct values on C in each instance in $\mathcal{I}^v$. Thus, the sets of the PFs in the second view linked to p_{11} and p_{12} are $\{p_{21}, p_{22}, p_{23}\}$, $\{p_{21}, p_{22}, p_{24}\}$, $\{p_{21}, p_{22}, p_{25}\}$, $\{p_{23}, p_{24}\}$, $\{p_{23}, p_{25}\}$, or $\{p_{24}, p_{25}\}$. As a result, in all instances in $\mathcal{I}^v$, the possible sets of values on D associated with a_1 are $\{d_1, d_2\}$, $\{d_1, d_2, d_3\}$, $\{d_1, d_2, d_4\}$, $\{d_2, d_3\}$, $\{d_2, d_4\}$, and $\{d_3, d_4\}$. Clearly, by the definition of association cover, the problem of finding association cover of size less than k from these sets is that of finding set cover on these sets of size less than k that is known to be NP-hard [5].

FD:$B \rightarrow C$
$\Pi_{A,B}(baseTbl)$

	A	B
p_{11}	a_1	b_1
p_{12}	a_1	b_2
p_{13}	a_2	b_3
p_{14}	a_2	b_4

Secret:(A, D)
$\Pi_{C,D}(baseTbl)$

	C	D
p_{21}	c_1	d_1
p_{22}	c_1	d_2
p_{23}	c_2	d_2
p_{24}	c_3	d_3
p_{25}	c_4	d_4

Figure 6: An example for the minimal association cover due to FD.

For this particular case, we can use an approximate checking method. The basic idea is that if, for each value a on ID, there exists k base table instances such that a is associated with a set of distinct values on P in each instance, then the view set does not violate k-anonymity. More formally, the following theorem holds. Indeed, this theorem is general enough to be applied to other subcases. Here A_I^a denotes the set of all associations with value a on ID that are in $S(I)$.

Theorem 5 *Given a view set v, for each value a on ID, if there exists k instance $I_1 ... I_k \in \mathcal{I}^v$ such that $\forall I_i, I_j \in \{I_1 ... I_k\}$, where $i \neq j$, $A_{I_i}^a \cap A_{I_j}^a = \emptyset$, then v does not violate k-anonymity.*

We take the example in Figure 6 to explain this approximate method. We want to check the violation of 2-anonymity and only consider the value a_1 on A in the example, which has to be associated with two distinct values on C. By the above theorem, we need to find 2 instances with disjoint sets of values on D that a_1 is associated with. In one instance, we can let c_1 and c_2 be associated with a_1 so that the set $\{d_1, d_2\}$ is associated with a_1. In another instance, we can choose c_3 and c_4 to be associated with a_1 so that $\{d_3, d_4\}$ is associated with a_1. Since $\{d_1, d_2\} \cap \{d_3, d_4\} = \emptyset$, the view set does not violate 2-anonymity.

Here we analyze the approximate ratio of this method. Let l be the least size of minimal association covers. In general, if we can only find the l' disjoint sets of associations in instances, where $l' < k$, then it may or may not violate k-anonymity. Since we cannot find any other instance without any association in

the union of these l' sets, the union is an association cover. Therefore, the least size of the association covers is less than or equal to the size of the union, hence $l' \leq l \leq (l' * max)$, where max is the maximum size among the disjoint sets of associations. For this example, let m be the maximum number of the distinct values on B each value on A is associated with and n is the maximum number of the distinct values on C each value on D is associated with. Since $max \leq (m * n)$, the approximate ratio is $1/(m * n)$.

If a given two-view set with a single FD has selection conditions, then "common attributes" includes the attributes appearing in the selection conditions, and the above study is still applicable. That is, if there are not common attributes, then there exists a link between any two PFs in the two views, and the FD does not affect the links between PFs but may affect minimal association cover.

6.3 *ID* as Key

In practice, it is usual that *ID* is the key of *baseTbl* and there are no other FDs except the FDs involving the key. Under this situation, checking is tractable in some subcases.

Because of the FD from *ID* to P, there is no instance in $\mathcal{I}^v$ such that a value on *ID* is associated with more than one values on P. Thus, we only need to check how many values on P each value on *ID* can be associated with.

6.3.1 The projection-only subcase

We first consider the projection-only subcase. Given a view set v, we construct a graph G as follows. Each view v_i in v is mapped to a node, denoted by $N(v_i)$, in G. If two views v_i and v_j have common attributes, then there exists an edge between $N(v_i)$ and $N(v_j)$. A *join path* is a path from $N(v_{ID})$ to $N(v_P)$ without passing any edge more than once. We only consider the subcase where there is one and only one join path. If there is no join path, checking is simple, since each value on *ID* in v_{ID} can be associated with any value on P in v_P, and hence checking is to simply count how many distinct values on P in v_P.

Proposition 8 *Given G constructed by v, where G has only one join path, if an association α exists in $\Pi_{ID,P}(J)$, then there exists instance I in $\mathcal{I}^v$ such that α is in $\Pi_{ID,P}(I)$.*

Let $\bar{A}$ be the set of attributes appearing in the views in the join path. Then the result of the natural join through the join path is a set of PFs upon $\bar{A}$, denoted by J. Since the following proposition holds, to count how many values on P can be associated with each value on *ID*, we can just perform natural join on the views through the join path and count those associations in the join result.

For example, we have a view set with 3 views $(\Pi_{Name,Job}, r_1)$, $(\Pi_{Job,Salary}, r_2)$ and $(\Pi_{Salary,Problem}, r_3)$ on the base table in Figure 1. And we know that $Name$ is the key. In order to check for k-anonymity violation, we can just perform $r_1 \bowtie r_2 \bowtie r_3$ to see in the join result how many values on $Problem$ in r_3 can be associated with each value on $Name$ in r_1, respectively.

6.3.2 The subcase with selection conditions

On extending this subcase to that with selection conditions, we still may be able to perform checking by *join* through the join path. The only difference is that a join between two views is not only a natural join considering selection conditions. Thus, we concentrate on two-view sets with selection conditions to demonstrate the checking method in this subcase.

We take the example in Figure 7 to explain the checking procedure. The checking is equivalent to determining the possible links between the PFs. We perform the following procedure. First, we apply the method in Section 5 to get the links L between the PFs disregarding the FDs, which is W in Section 5. We construct a bipartite graph G as Figure 7 from L. Two sets of nodes are mapped to the PFs in two views v_1 and v_2, respectively. If there is a link in L between p_1 in v_1 and p_2 in v_2, then there is an edge between the two corresponding nodes.

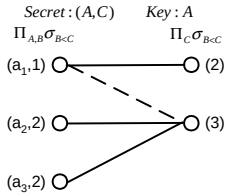


Figure 7: An example for the bipartite graph for links.

Then we need to remove the links that are not allowed due to the FDs, which actually is to impose FD on W. The link represented by the dashed line in the figure is removed in this step. The implementation of this step is as follows. Since each possible base table instance yields a set of links between r_1 and r_2, a subgraph m of G. Because A is the key and two views have the same condition, each node mapped to a PF in the first view has to be associated with one and only one node corresponding to a PF in the second view. Therefore, m has the following properties: (1) Each node in G is incident to at least one edge in m; (2) no two edges are in m with the same endpoint on a node mapped to a PF in the first view. In Figure 7, there is only one subgraph m, which is the links in solid lines.

Each link must appear in such a subgraph m. Thus, the problem is transformed into checking whether an edge e exists in such a subgraph m. We can do this by checking whether e exists in a complete matching of G, which can be seen from the properties of m. To determine e being in a complete matching, we can

check whether a complete matching can be found in $G - N(e)$, where $N(e)$ are the two nodes incident to e. Since finding a complete matching is polynomial time in the size of bipartite graph, checking is polynomial time in the number of PFs in v.

7 Conservative Checking

The previous section shows that accurate checking for k-anonymity violation in many cases is intractable. Even though the data complexity of checking in other cases is polynomial time, it may be too costly to be practical, especially for view sets with large number of tuples. Thus, we introduce a conservative checking method that only checks necessary condition for k-anonymity violation. We call a checking method "conservative" if it always catches k-anonymity violation, but it may make mistakes when a view set actually does not violate k-anonymity.

All the necessary conditions in this section are based on the idea "indistinguishable characteristics" of the values in secret associations. For example, consider the three views in Figure 8. In the view containing *Problem*, the attribute *Salary* could be used to link *Problem* to *Name*, since there is an FD $Job \rightarrow Salary$. The two PFs containing *Cold* and *Obesity* have the same value on *Salary*. As a result, if in an instance, a value on *Name*, e.g., *George*, is associated with only *Cold* (*Obesity*, respectively), then there exists another instance such that it is associated with *Obesity* (*Cold*, respectively). Due to $Name$ is the key, since *George* has to be associated with one problem at a time, either $(George, Name)$ or $(George, Obesity)$ alone cannot be an association cover. Thus, both values cannot incur the violation of 2-anonymity. In contrast, *HIV* is unique, hence may incur the violation of 2-anonymity. In this sense, we intend to seek the functions catching such characteristics and find enough different values on P with the same characteristics regarding associations with values on *ID*.

Key:$Name$ FD:$Job \rightarrow Salary$

$\Pi_{Name,Zip}(baseTbl)$

Name	Zip
George	20001
John	20001
Bill	20002

$\Pi_{Zip,Job}(baseTbl)$

Zip	Job
20001	Manager
20001	Manager
20002	Lawyer

$\Pi_{Salary,Problem}(baseTbl)$

Salary	Problem
100K	Cold
100K	Obesity
150K	HIV

Figure 8: An example for the characteristics of join. Cold is the same as Obesity while HIV is unique w.r.t. characteristics base on the PFs containing Problems.

From the above example, we can see that this is a kind of rough and fast conservative checking method. We believe this method is practical. Furthermore, this method makes it possible to tradeoff accuracy with efficiency by introducing different levels of functions for such characteristics. Basically, the more conservative the checking method, usually the faster the checking. In addition, this method suggests a way to anonymize the view sets that violate k-anonymity. That is, generalize and suppress the appropriate attribute values to make the characteristics of the enough number of values on P or *ID* be the same. We do not go into this direction in this paper.

7.1 General Method

Let $\mathcal{ID}^D$ denote the set of all allowable values on *ID* in the base table schema D and $\mathcal{P}^D$ denote those on P. We introduce the concept of *signature function* to measure the characteristics of the values on *ID* and P.

Definition (**Signature Function**) Given v, a *signature function* f_v is a mapping $\mathcal{ID}^D \times \mathcal{P}^D \rightarrow R$, where R is a infinite set. Give $a \in \mathcal{ID}^D$ and $b \in \mathcal{P}^D$, $f_v(a, b)$ is called the signature of b for a.

A signature function can be looked as a set of customized queries on v. The argument of each query is a pair (a, b) in $\mathcal{ID}^D \times \mathcal{P}^D$. As the values of a function are not related to a or b, the function can be *abbreviated* to $f(b)$ or $f(a)$. In this paper, we are only interested in the particular class of signature functions, *symmetric signature function*. Before giving its definition, we introduce the following concept. Here $S_a(I)$ denotes the set of all associations in $S(I)$ that have the same value a on *ID*.

Definition (**Symmetric Values**) Given a view set v, a value a in $\mathcal{ID}^D$ and values b_1, b_2 in $\mathcal{P}^D$, b_1 and b_2 are *symmetric* for a, if the following condition is satisfied: $\forall I \in \mathcal{I}^v$, if exactly one of (a, b_1) and (a, b_2) is in $S_a(I)$, there exists I' in $\mathcal{I}^v$ such that $S_a(I') = (S_a(I) - \{(a, b_1), (a, b_2)\}) \cup (\{(a, b_1), (a, b_2)\} - S_a(I))$.

Proposition 9 *Symmetry between values is transitive.*

This proposition says that if b_1 is symmetric to b_2 for a and b_2 is symmetric to b_3 for a, then b_1 is symmetric to b_3 for a.

Definition (**Symmetric Signature Function**) Given a view set v, a signature function f is *symmetric* if the following condition is satisfied: $\forall a \in \mathcal{ID}^D$ and $\forall b_1, b_2 \in \mathcal{P}^D$, if $f(a, b_1) = f(a, b_2)$, then b_1 and b_2 are symmetric for a.

For the example in Figure 8, for a value b on *Problem*, we define a signature function that returns the value on *Salary* associated with b. For example, $f_v(Bill, Cold) = 100K$, which can be abbreviated to

$f_v(Bill)$. *George* or *John* can be associated with either *Cold* or *Obesity* in possible instances; *Bill* is not associated with any one of them in all instances. By the definition of symmetry, *Cold* and *Obesity*, which have the same signature, are symmetric for all values on *Name*. Thus, this function is symmetric.

We can utilize symmetric signature functions to check if two values are symmetric. Since symmetry between values is transitive, for two values b_1 and b_n in $\mathcal{P}^D$, if there exist symmetric signature functions f_1, f_2, ..., f_{n-1} and b_2, b_3, ..., b_{n-1} in $\mathcal{P}^D$ such that $f_1(a, b_1) = f_1(a, b_2)$, $f_2(a, b_2) = f_2(a, b_3)$, ..., $f_{n-1}(a, b_{n-1}) = f_{n-1}(a, b_n)$, then b_1 and b_n are symmetric for a. Thus, we can introduce many different symmetric signature functions into the checking for symmetry between values.

Theorem 6 *Given a view set v and a value a in $\mathcal{ID}^D$, v does not violate k-anonymity for a, if there exists I in $\mathcal{I}^v$, the following condition is satisfied: For each association (a, b) in $S(I)$, there exists a set of $k-1$ distinct values b_i such that b_i is symmetric to b for a and (a, b_i) is not in $S(I)$.*

Procedure *Checking k-anonymity for a view set*
Input: v, $S = \Pi_{ID,P}$, integer k
and a set F of symmetric signature functions
Output: True (possible violation) or False

```
For each (a, b) in S(baseTbl), a and b are in v
    Let U = {b}
    For i=1 to k − 1
        If ∃b_i ∉ U, ∃f ∈ F f(a, b_i) = f(a, b_j),
        b_j ∈ U and (a, b_i) ∉ S(baseTbl)
            Let U = U ∪ b_i
        Else Return true;
Return false
```

Figure 9: A general procedure for conservative checking for k-anonymity

Theorem 6 is the basis of our conservative checking method. We may choose the current base table as I in the theorem. Usually, we do not need to check k-anonymity for all values on *ID* and P in *baseTbl*, but those that appear in v. Because in most cases the minimal association covers are composed of the values in v. Therefore, we introduce a general checking procedure as Figure 9. The basic idea is that for each association (a, b) in $S(baseTbl)$, check whether we can find $k-1$ distinct values that are symmetric to b for a by a given set of symmetric signature functions.

7.2 Symmetric Signature Functions

Now we suggest some symmetric signature functions in the projection-only subcase. It is easy to extend those signature functions to the subcase with selection conditions by incorporating the attributes in selection conditions. We still suppose there are one view v_{ID} containing *ID* and another view v_P containing P. We use $\mathcal{ID}^v$ to denote the set of all values on *ID* in v and $\mathcal{P}^v$ to denote those on P.

Given a view v_1, a view v_2 is a linkable view for v_1, if (1) there exist common attributes between v_1 and v_2, or an FD across v_1 and v_2; Or (2) there exists view v_3 such that v_2 is a linkable view for v_3, and v_3 is a linkable view for v_1, which means that it is transitive. Clearly, v_1 is a linkable view for itself. We let L_C denote the set of all the attributes in v_P that appear in a linkable view for v_{ID}; L_F the set of all the attributes in v_P that have FD relationship with an attribute in a linkable view for v_{ID}, excluding the attribute P. Indeed, L_C and L_P collect all the attributes that affect the characteristics of the values on P regarding associations. For example, in the example of Figure 8, *Salary* is in L_F; and if *Zip* were contained in the third view, *Zip* should be in L_C.

Definition (**Complete Signature Function**) The complete signature function $f_c^P(b)$ is the result of the query $\Pi_{L_C \cup L_F} \sigma_{P=b}(v_P)$ without eliminating duplicates.

For the example in Figure 8, for any value on *Name*, the complete signature of *Cold* is $\{(100K)\}$ on *Salary*, which is the same as that of *Obesity*.

Proposition 10 *The complete signature function is symmetric.*

Complete signatures are not related to values on *ID*. Thus, we do not need to compute signatures for each value on *ID*. As a result, the algorithm in Figure 9 can be simplified greatly. Furthermore, complete signatures are simple relational queries, hence can be easily evaluated. But the complete signature function reflects a very necessary condition for k-anonymity violation, since for a value b on P, it conservatively records the projection values of all tuples that b appears in upon the set of all attributes that may affect links between PFs.

Secret:(A, B)

FD:$A \to D$

$\Pi_{A,C}$

A	C
a_1	c_1
a_1	c_2
a_2	c_1
a_2	c_2

$\Pi_{C,D,B}$

C	D	B
c_1	d_1	b_1
c_2	d_1	b_2
c_1	d_2	b_3
c_2	d_2	b_4

(a)

FD:$A \to B$

$\Pi_{A,C}$

A	C
a_1	c_1
a_2	c_1
a_3	c_2
a_4	c_2

$\Pi_{C,B}$

C	B
c_1	b_1
c_1	b_2
c_2	b_1
c_2	b_3

(b)

Figure 10: Examples for symmetric signature functions.

There are some other symmetric signature functions, which can be more complicated but reflect less necessary conditions for k-anonymity violation. For the sake of space, we list only two by example. One

makes two values b_1 and b_2 have the same signature, if for the PFs containing b_1 and b_2, the projection values on L_c are the same, and the projection values on L_F have the same complete signature. In Figure 10(a), b_1 and b_3 have the same signature because the values on C are the same and the values on D, d_1 and d_2, have the same complete signature. The other covers the example in Figure 10(b). Such a signature function, which is related to values on ID, causes that $f(a_1, b_1) = f(a_1, b_2) = \{(c_1)\}$ and $f(a_3, b_1) = f(a_3, b_3) = \{(c_2)\}$.

8 Related Work

Analysis of information disclosure with views is not new (see, e.g., [8, 3, 4]). Miklau and Suciu [8] were perhaps the first to apply perfect secrecy to the analysis of information disclosure, and mainly concentrated on the conditions of absolute safety. In a followup paper [3], Dalvi et.al. suggested a method to calculate the disclosure. In [4], Deutsch and Papakonstantinou studied whether or not an extra view discloses more information than the existing views with respect to a secret view. However, these methods are based on probability model and are complex and difficult to be applied practically. Moreover, the authors did not measure disclosure at the tuple level. This paper applied k-anonymity, which is more intuitive, to measure disclosure at the tuple level.

Previous works about k-anonymity concentrated on how to gain k-anonymity by modifying the data in a single view or table [11, 12, 7, 6]. The authors did not study how to check releasing views for k-anonymity violation. In addition, k-anonymity considered is between the given Quasi-ID and sensitive attributes. In this paper, the checking methods implicitly gave a way to identify such quasi-IDs. There are other works, which are not about k-anonymity, on how to protect secrecy or privacy by modifying data in a single view. One of them is privacy in data mining, first proposed by Agrawal and Srikant [1]. In contrast, we concentrated on checking for k-anonymity violation.

Another related field is inference control. Authors studied the information disclosure resulted by FDs or other constraints at the tuple level. One of recent works is [2]. Unlike in this paper, the method in [2] checks whether we can infer sensitive attribute values of a tuple from other tuples based on the known constraints, but does not check the inferences by linking tuples based on view definitions.

9 Conclusions

In this paper, we applied k-anonymity to measure the information disclosure of multiple releasing views with respect to secret associations. We showed that checking is polynomial time when the base table does not have FDs, while checking is highly complex if the given view set does. In fact, the general checking is hard even for a two-view set with a single FD. However, it is still practical to tackle the checking problem for the case with FDs. Indeed, we presented some representative subcases where checking is easy or can be approximated. In order to avoid the complex checking for a given view set, one solution is to make the releasing view set in one of the tractable subcases. That is, choose the queries for releasing data such that checking is not hard. Another solution is to use our conservative checking method.

For future work, we will perform some experiments on real data to verify the checking methods for k-anonymity violation. Furthermore, in this paper, we assumed view sets do not preserve duplicates. Views with duplicates may disclose more information, and checking methods need be revised. Finally, we concentrated on views having only selection and projection. Checking methods need to be extended to other kinds of views such as those using joins.

References

[1] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *SIGMOD*, 2000.

[2] A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *TKDE*, 12(6), 2000.

[3] N. Dalvi, G. Miklau, and D. Suciu. Asymptotic conditional probabilities for conjunctive queries. In *ICDT*, 2005.

[4] A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In *ICDT*, 2005.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, San Francisco, 1979.

[6] R. J. B. Jr. and R. Agrawal. Data privacy through optimal k-anonymization. In *ICDE*, 2005.

[7] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *PODS*, 2004.

[8] G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *SIGMOD*, 2004.

[9] P. Revesz. *Introduction to constraint databases*. Springer-Verlag, 2002.

[10] S. Rizvi, A. O. Mendelzon, S. Sudarshan, and P. Roy. Extending query rewriting techniques for fine-grained access control. In *SIGMOD*, 2004.

[11] P. Samarati. Protecting respondents' identities in microdata release. *TKDE*, 13(6):1010–1027, Nov. 2001.

[12] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *PODS*, 1998.

[13] C. Umans. *Approximability and Completeness in the Polynomial Hierarchy*. PhD thesis, University of California, Berkeley, Fall 2000.