Indistinguishability: the Other Aspect of Privacy*

Chao Yao[1]**, Lingyu Wang[2], Sean X. Wang[3], and Sushil Jajodia[1]

[1] Center for Secure Information Systems
George Mason University
{cyao,jajodia}@gmu.edu
[2] CIISE, Concordia University
wang@encs.concordia.ca
[3] Department of Computer Science
The University of Vermont
xywang@cs.uvm.edu

Abstract. Uncertainty and indistinguishability are two independent aspects of privacy. Uncertainty refers to the property that the attacker cannot tell which private value, among a group of values, an individual actually has, and indistinguishability refers to the property that the attacker cannot see the difference among a group of individuals. While uncertainty has been well studied and applied to many scenarios, to date, the only effort in providing indistinguishability has been the well-known notion of k-anonymity. However, k-anonymity only applies to anonymized tables. This paper defines indistinguishability for general situations based on the symmetry among the possible private values associated with individuals. The paper then discusses computational complexities of and provides practical algorithms for checking whether a set of database views provides enough indistinguishability.

1 Introduction

In many data applications, it's necessary to measure privacy disclosure in released data to protect individual privacy while satisfying application requirements. The measurement metrics used in prior work have mainly been based on uncertainty of private property values, i.e., the uncertainty what private value an individual has. These metrics can be classified into two categories: non-probabilistic and probabilistic. The non-probabilistic metrics are based on whether the private value of an individual can be uniquely inferred from the released data [1, 20, 7, 17, 5, 16] or whether the cardinality of the set of possible private values inferred for an individual is large enough [26, 27]. The probabilistic metrics are based on some characteristics of the probability distribution of

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

** Part of work of this author was done while visiting the University of Vermont.

the possible private values inferred from the released data $[3, 2, 10, 9, 15, 4]$ (see Section 4 for more details).

However, uncertainty is only one aspect of privacy and it alone does not provide adequate protection. For example, we may reveal employee John's salary to be in a large interval (say, $100K$ to $300K$ annually). There may be enough uncertainty. However, if we also reveal that the salaries of all other employees are in ranges that are totally different from John's range (say, all are subranges of $50K$ to $100K$), then John's privacy may still be violated. As another example, suppose from the released data we can infer that all patients in a hospital may only have $Cold$ or $SARS$ except that $John$ may have $Cold$ or $AIDS$. Even though the uncertainty of $John$'s sickness has the same "magnitude" as that of the other patients, $John$ may still feel his privacy is violated, since he is the only one who possibly has $AIDS$.

To adequately protect privacy, we need to consider the other aspect, namely, *indistinguishability*. Indeed, the privacy breach in the above examples can be viewed as due to the fact that from the released data, an individual is different from all other individuals in terms of their possible private values. In other words, the examples violate a privacy requirement, namely, the "protection from being brought to the attention of others" [11]. What we need is to have each individual belong to a group of individuals who are indistinguishable from each other in terms of their possible private values derived from the released data. In this way, an individual is hidden in a crowd that consists of individuals who have similar/same possible private values. For instance, in the above salary example, to protect John's privacy, we may want to make sure that attackers can only derive from the released data that a large group of employees have the same range as John's for their possible salaries.

Uncertainty and indistinguishability are two independent aspects for providing privacy; one does not imply the other. From the above examples, we can see that uncertainty cannot ensure good indistinguishability. Likewise, good indistinguishability cannot ensure enough uncertainty. For instance, if in the released data many employees have the same single possible salary value, then these employees are indistinguishable from each other in terms of their salaries, but there is not enough uncertainty to protect their privacy (all their salaries are the same and revealed!).

Our idea of indistinguishability is inspired by the notion of k-anonymization [24, 25, 21, 14, 18] as it can be viewed as a generalization of anonymization. The idea of k-anonymization is to recode, mostly by generalization, publicly available quasi-IDs in a single released table, so that at least k individuals will have the same recoded quasi-IDs. (Quasi-IDs are values on a combination of attributes that can be used to identify individuals through external sources [24, 25].) In our view, this is an effort to provide indistinguishability among k individuals, since the recoding makes the individuals indistinguishable from each other. (As noted above, indistinguishability does not guarantee uncertainty. This is also true for k-anonymization, which is illustrated by the improvement reported in [19]. The authors impose an additional requirement on anonymization, namely, by

requiring diverse private values among the tuples with the same recoded quasi-ID, in order to achieve, in our view, both indistinguishability and uncertainty.)

While k-anonymity is an interesting notion, it only applies to anonymized tables. In this paper, we define two kinds of indistinguishability, and the corresponding privacy metrics, that can be applied to general situations, including anonymized tables and relational views. We show that k-anonymization is a special case of one kind of indistinguishability under a certain assumption (see Section 2.3).

Both notions of indistinguishability introduced in this paper are based on certain symmetry between individuals and their private values in the released data. More specifically, the first definition requires symmetry for all possible private values while the second definition bases on symmetry referring only to certain subsets of possible private values. With the two kinds of indistinguishability defined, we turn to study the problem of deciding whether a set of database views provides enough indistinguishability. We study the computational complexity as well as practical algorithms. We focus on checking for indistinguishability since checking for uncertainty has been extensively studied [1, 7, 17, 5, 26, 16, 27].

We summarize the contributions of this paper as follows. (1) We identify indistinguishability as a requirement for privacy in addition to uncertainty, provide formal definitions of different kinds of indistinguishability, and study their properties. (2) We analyze the computational complexity and introduce practical checking methods for deciding whether a set of database views provides enough indistinguishability.

The rest of paper is organized as follows. We give formal definitions of indistinguishability and privacy metrics in Section 2. We then focus on checking database views against these privacy metrics in Section 3. In Section 4 we review the related work. Finally, we conclude with a summary in Section 5.

2 Indistinguishability

2.1 Preliminaries

In this paper, we consider releasing data from a single private table Tbl with schema D. The attributes in D are partitioned into two sets, B and P. The set B consists of the public attributes; P consists of the private attributes. For simplicity and without loss of generality, we assume P only has one attribute.

We assume that the projection on B, $\Pi_B(Tbl)$, is publicly known. In the salary example, this means that the list of employees is publicly known. We believe this assumption is realistic in many situations. In other situations where this is not true, we may view our approach as providing a conservative privacy measure.

Given a relation r_B on B, we will use $\mathcal{I}^B$ to denote the set $\{I|\Pi_B(I) = r_B\}$, i.e., the set of the relations on D whose B-projection coincides with r_B. The domain of P is denoted by $Dom(P)$. A tuple of an instance in $\mathcal{I}^B$ is denoted by t or (b, p), where b is in $\Pi_B(Tbl)$ and p is in $Dom(P)$. The set $\mathcal{I}^B$ corresponds to all possible private table instances by only knowing $\Pi_B(Tbl)$.

Furthermore, we assume B is a key in D, which means that each composite value on B appears at most once in the private table. We also assume B is a quasi-ID, and hence, the tuples in Tbl describe associations of the private attribute values with individuals. (Recall that a quasi-ID is a combination of attribute values that can be used to identify an individual.) Such associations are the private information to be protected.

In Figure 1, our running example is shown. The public attributes in B are Zip, Age, $Race$, $Gender$, and $Charge$. We use $t_1, ..., t_{12}$ to denote the tuples in the table. By the assumption that B is a quasi-ID, $t_i[B]$ identifies a particular individual for each i. In the sequel, we use $t_i[B]$ and the individual identified by $t_i[B]$ interchangeably. The private attribute is $Problem$. Here, $Problem$ is drawn from a finite discrete domain. (In general the private attribute also can be drawn from an infinite or a continuous domain; but it should not be difficult to extend our study to infinite discrete or continuous domains).

We assume that the data in Tbl are being released with a publicly-known function M. We also use v to denote the result of $M()$ on the private table, i.e., $v = M(Tbl)$. Examples of function $M()$ include an anonymization procedure, and a set of queries (views) on a single table on D.

	Zip	Age	Race	Gender	Charge	Problem
t_1	22030	39	White	Male	1K	Cold
t_2	22030	50	White	Male	12K	AIDS
t_3	22030	38	White	Male	5K	Obesity
t_4	22030	53	Black	Male	5K	AIDS
t_5	22031	28	Black	Female	8K	Chest Pain
t_6	22031	37	White	Female	10K	Hypertension
t_7	22031	49	Black	Female	1K	Obesity
t_8	22031	52	White	Male	8K	Cold
t_9	22032	30	Asian	Male	10K	Hypertension
t_{10}	22032	40	Asian	Male	9K	Chest Pain
t_{11}	22033	30	White	Male	10K	Hypertension
t_{12}	22033	40	White	Male	9K	Chest Pain

Fig. 1. A patient table (Tbl)

	Zip	Problem
t_9	22032	Hypertension
t_{10}	22032	Chest Pain
t_{11}	22033	Hypertension
t_{12}	22033	Chest Pain

Fig. 2. A released view $\Pi_{Zip,Problem}(Tbl)$ $\sigma_{Zip='22032' or '22033'}(Tbl)$ provides 2-SIND.

2.2 Symmetric Indistinguishability

As $v = M(Tbl)$ is released, we denote by $\mathcal{I}^v$ the subset of possible instances in $\mathcal{I}^B$ that yield v. We introduce the definition of indistinguishability based on $\mathcal{I}^v$.

Definition 1. *(***Symmetric Indistinguishability***) Given a released data v and two tuples b_i and b_j in $\Pi_B(Tbl)$, we say b_i and b_j are* symmetrically Indistinguishable *w.r.t. v if the following condition is satisfied: for each instance I in $\mathcal{I}^v$ containing (b_i, p_i) and (b_j, p_j), there exists another instance I' in $\mathcal{I}^v$ such that $I' = (I - \{(b_i, p_i), (b_j, p_j)\}) \cup \{(b_i, p_j), (b_j, p_i)\}$.*

We abbreviate Symmetric Indistinguishability as *SIND*. This definition requires that for each possible instance in $\mathcal{I}^v$, if two symmetrically indistinguishable B tuples swap their private values while keeping other tuples unchanged, the resulting new instance can still yield v. In the sequel, we say *two B tuples $t_1[B]$ and $t_2[B]$ can swap their private values in an instance*, or simply $t_1[B]$ *swaps with* $t_2[B]$, if the resulting instance can still yield v.

Note that such a swap is required for all the instances yielding v, hence this definition is in terms of v, not the current table Tbl (although we used the projection $\Pi_B(Tbl)$ in the definition, this projection is not Tbl itself and is assumed publicly known). In other words, to be SIND is to be able to swap their private values in all the possible instances, including Tbl.

For example, consider the released view v in Figure 2 on the table in Figure 1. The two B tuples $t_9[B]$ and $t_{10}[B]$ are SIND, because they can swap their $Problem$ values in any instance that yields v while still yielding the same v. Similarly, the two B tuples $t_{11}[B]$ and $t_{12}[B]$ are also SIND. However, $t_9[B]$ and $t_{11}[B]$ are not SIND, even though they have the same $Problem$ value $Hypertension$ in the current private table. To show this, consider an instance obtained by swapping the $Problem$ values of t_9 and t_{10} in Tbl (while other tuples remain unchanged). So now t_9 has $ChestPain$ while t_{10} has $Hypertension$. Denote the new instance Tbl'. Clearly, Tbl' also yields the view v. However, in Tbl', if we swap the $Problem$ values of t_9 (i.e., $ChestPain$) with that of t_{11} (i.e., $Hypertension$), then both t_9 and t_{10} will have $Hypertension$. Therefore, the new instance obtained from Tbl' does not yield v, and hence t_9 and t_{11} are not SIND.

The definition of SIND requires a complete symmetry between two B tuples in terms of their private values. The sets of possible private values of the SIND tuples are the same, because in each possible instance two SIND B tuples can swap their private values without changing the views. Furthermore, the definition based on swapping makes SIND between two B tuples independent on other B tuples. That is, even if attackers can guess the private values of all other B tuples, they still cannot distinguish between these two B tuples because the two B tuples still can swap their private values without affecting the views.

We can also use a probability model to illustrate the indistinguishability by SIND. If we assume each B tuple has the same and independent *a priori* distribution over its private values, then we can easily prove that the two B tuples have the same *a posteriori* distribution over their private values after data released, due to complete symmetry in terms of their private values.

The binary relation SIND is *reflexive*, *symmetric* and *transitive*. That is, SIND is an *equivalence* relation. It is easy to see that it is reflexive and symmetric. We prove the transitivity as follows. If a B tuple b_1 can swap with another B tuple b_2 and b_2 can swap with b_3, then b_1 can swap with b_3 by the following steps: b_1 swaps with b_2; b_2 swaps with b_3; b_2 swaps with b_1; by the definition of SIND, the final instance still yields v.

Thus, all the B tuples that are indistinguishable from each other form a partition of the B tuples. Each set in the partition, which we call a *SIND set*, is the "crowd" that provides individual privacy. The sizes of these crowds reflect how much protection they give to the individuals in the crowd. So we have the following metric.

Definition 2. *(k-**SIND**) Given a released data v, if each SIND set has a cardinality of at least k, we then say v provides k-*SIND*.*

2.3 Relationship with k-Anonymity

In this subsection, we discuss the relationship between k-SIND and k-anonymity. In the k-anonymity literature (e.g., [24, 25, 21, 14, 18]), the released data is an anonymized table. Anonymization is a function from quasi-IDs to recoded quasi-IDs, and the anonymization process (the function M in Section 2.1) is to replace quasi-IDs with recoded quasi-IDs. We assume that the anonymization algorithm and the input quasi-IDs are known. In fact, we make a stronger assumption, called "mapping assumption", which says that (1) each quasi-ID maps to one recoded quasi-ID and (2) given a recoded quasi-ID, attackers know which set of quasi-IDs map to it.

As an example, there is a table and an anonymized table as the following, respectively. The tuples on $(Zip, Race)$ are quasi-IDs. Under the mapping assumption, attackers know which quasi-ID maps to which recoded quasi-ID. For instance, (22031, $White$) maps to (2203$*$, $*$) but not (220$**$, $White$). (In contrast, without the mapping assumption, only from the anonymized table, (22031, $White$) may map to either (2203$*$, $*$) or (220$**$, $White$).)

Zip	**Race**	**Problem**
22021	White	Cold
22031	White	Obesity
22032	White	AIDS
22033	Black	Headache

Zip	**Race**	**Problem**
220$**$	White	Cold
220$**$	White	Obesity
2203$*$	$*$	AIDS
2203$*$	$*$	Headache

Under the above assumption, we have the following conclusion about the relationship between k-SIND and k-anonymity. Here the attributes of quasi-IDs are assumed to be exactly the public attributes B.

Proposition 1. *Under the mapping assumption, if an anonymized table v provides k-anonymity, where $k \geq 2$, then v provides k-SIND.*

Intuitively, if v provides k-anonymity, then at least k quasi-IDs map to each recoded quasi-ID in v. In any instance yielding v, suppose two quasi-IDs b_1 and b_2 map to the same recoded quasi-ID. Then swapping the private values of b_1 and b_2 in the original table gives an instance yielding the same v. Therefore, v provides k-SIND.

By definition, k-anonymity is applicable only to a single anonymized table, but not to other kinds of released data such as multiple database views.

2.4 Restricted Symmetric Indistinguishability

Since SIND requires symmetry in terms of all possible private values, it is a rather strict metric. We define another metric based on the symmetry in terms of not all possible private values but only a subset that includes the actual private values in the current private table. If B tuples are symmetric in terms of this subset of private values, even though they are not symmetric in terms of other values, we may still take them as indistinguishable. The intuition here is that we intend to provide more protection on the actual private values.

We associate each B tuple with a set of private values including its current private value. These sets form a collection. More specifically, we call a collection $\mathcal{P}$ of $Dom(P)$ value sets P_1, ..., P_n a *private value collection*, where $n = |\Pi_B(Tbl)|$ and $\Pi_B(Tbl) = b_1$, ..., b_n, if for each s, where $s = 1, ..., n$, $\Pi_P \sigma_{B=b_s}(Tbl) \in P_s$.

If two B tuples are symmetric w.r.t. a private value collection, then we take them as indistinguishable. More formally, we have the following definition. We abbreviate restricted symmetric indistinguishability as RSIND.

Definition 3. *(***RSIND***) Given a released data v on the current table Tbl and a private value collection P_1, ..., P_n, we say two B tuples b_i and b_j are RSIND w.r.t. P_1, ..., P_n if the following conditions are satisfied: (1) $P_i = P_j$ and (2) for each p_i in P_i and each p_j in P_j, if (b_i, p_i) and (b_j, p_j) are in an instance I in $\mathcal{I}^v$, I' is in $\mathcal{I}^v$ where $I' = (I - \{(b_i, p_i), (b_j, p_j)\}) \cup \{(b_i, p_j), (b_j, p_i)\}$.*

In this definition, unlike SIND, which swaps all possible private values, RSIND only swaps private values in a subset including the current private values. RSIND becomes SIND if $P_i = Dom(P)$ for each i.

For example, consider the two views in Figure 3 (see caption) on the table in Figure 1. From the view, we can deduce that in the private table Tbl, $t_1[B]$ cannot take *Obesity* but can take $Cold$ and $AIDS$, and $t_2[B]$ can take all the three problems. Clearly, $t_1[B]$ and $t_2[B]$ are not SIND. But there exists a private value collection P_1, ..., P_4 with $P_1 = P_2 = \{Cold, AIDS\}$ and $P_3 = P_4 = \{Cold, AIDS, Obesity\}$, we have $t_1[B]$ and $t_2[B]$ are RSIND w.r.t. this collection. Indeed, P_1 and P_2 are identical, and they both include the current private values of $t_1[B]$ and $t_2[B]$, $Cold$ and $AIDS$. In any instance yielding the views, if $t_1[B]$ and $t_2[B]$ have $Cold$ and $AIDS$, or $AIDS$ and $Cold$, respectively, then swapping their private values results in an instance yielding the same views.

	Problem
t_1	Cold
t_2	AIDS
t_3	Obesity
t_4	AIDS

(a)The first view

$t_1[B]$
$t_2[B]$
$t_3[B]$
$t_4[B]$

(b)The SIND partition

$t_1[B]$	$\{Cold, AIDS\}$
$t_2[B]$	$\{Cold, AIDS\}$
$t_3[B]$	$\{Cold, AIDS, Obesity\}$
$t_4[B]$	$\{Cold, AIDS, Obesity\}$

(c) The RSIND partition w.r.t. a collection

Fig. 3. Two released views $\Pi_{Problem}\ \sigma_{Zip='22030'}(Tbl)$ and $\Pi_{Problem}\ \sigma_{t_1 \text{ and } Problem='Obesity'}(Tbl) = \emptyset$.

Given a private value collection P_1, ..., P_n, RSIND is also a binary equivalence relation, hence induces a partition over the B tuples; and each set in the partition is called an RSIND set w.r.t. P_1, ..., P_n.

Definition 4. *(k-**RSIND**) Given a released data v, if there exists a private value collection $\mathcal{P}$ such that each RSIND set in the induced partition has a cardinality of at least k, we then say v provides k-*RSIND*.*

Obviously, if v provides k-SIND, we can let P_1, ..., P_n be the collection of all possible private values, i.e., $P_s = \{p | \exists I \in \mathcal{I}^v\ (b_s, p) \in I\}$, where $s = 1$, ..., n. Then each pair of SIND values are RSIND w.r.t. P_1, ..., P_n, hence the SIND partition is the RSIND partition w.r.t. P_1, ..., P_n. Clearly, the cardinality of each set in this RSIND partition is at least k since each set in the SIND partition is so. Thus, we have the following proposition. In Figure 3, the second view makes t_1 not have *Obesity* which others may have. The views do not provide 2-SIND, but do provide 2-RSIND.

Proposition 2. *k-SIND implies k-RSIND.*

From the definition of RSIND, we have the following conclusion. Given a set of tuples T in the current table Tbl, each private value collection w.r.t. which the B tuples in $\Pi_B(T)$ are RSIND from each other must include all of their current private values, $\Pi_P(T)$; $\Pi_B(T)$ are RSIND from each other w.r.t. $\Pi_P(T)$, if there exists a collection such that $\Pi_B(T)$ are RSIND from each other. More formally, we have the following proposition.

Proposition 3. *Given a private value collection $\mathcal{P} = P_0, P_1, ..., P_n$, released data v, and a set T of tuples in the current table, if the tuples in $\Pi_B(T)$ are RSIND from each other w.r.t. $\mathcal{P}$, then we have the following two facts. First, for each b_i in $\Pi_B(T)$, $\Pi_P(T) \subseteq P_i$. Second, for each b_i in $\Pi_B(T)$, if we replace P_i with $\Pi_P(T)$ to get a new private value collection $\mathcal{P}'$, then all the B tuples in $\Pi_B(T)$ are still RSIND w.r.t. $\mathcal{P}'$.*

Consider the example in Figure 3. $t_3[B]$ and $t_4[B]$ are RSIND w.r.t. the private value collection. Hence, in the collection, the corresponding sets of $t_3[B]$ and $t_4[B]$ are identical and have both their current private values, *Obesity* and $AIDS$. If we take their current private values as a collection, which means dropping *Cold* from their corresponding sets, $t_3[B]$ and $t_4[B]$ are still RSIND.

Proposition 3 implies the following property of RSIND. If the tuples in $\Pi_B(T)$ are RSIND from each other, then by Proposition 3, the tuples in $\Pi_B(T)$ are RSIND from each other w.r.t. $\Pi_P(T)$. By a repeated use of the definition of RSIND, for each set of tuple T' such that $\Pi_B(T') = \Pi_B(T)$ and the private values in T' is a permutation of the private values (with duplicates preserved) in T, we know there exists an instance I in $\mathcal{I}^v$ with $T \subseteq I$. This explains why we say these tuples are indistinguishable in terms of the current private values.

For example, consider the SIND partition of Figure 3(b) as an RSIND partition (note again that there are many RSIND partitions with difference private value collections and the SIND partition is one of them). We have that $t_2[B]$, $t_3[B]$ and $t_4[B]$ are RSIND from each other w.r.t. $P_2 = P_3 = P_4 = \{Obesity, AIDS\}$. Then for each of the three different (repeated) permutations of $t_2[B]$, $t_3[B]$, and $t_4[B]$ with *Obesity*, $AIDS$ and $AIDS$ values (i.e., $\langle (t_2[B], Obesity), (t_3[B], AIDS), (t_4[B], AIDS) \rangle$, $\langle (t_2[B], AIDS), (t_3[B], Obesity), (t_4[B], AIDS) \rangle$,

and $\langle(t_2[B], AIDS), (t_3[B], AIDS), (t_4[B], Obesity)\rangle$), there exists at least one instance in $\mathcal{I}^v$ that contains that permutation.

The size of each set in a private value collection matters in measuring privacy disclosure, which is not reflected in k-RSIND. Generally, the more P values in the collection, the better indistinguishability we achieve since we ignore the fewer P values that may make B tuples distinguishable. Also, more private values may mean better uncertainty. However, in this paper, we are not pursuing this direction.

3 Checking Database Views

In this section, we focus on checking released data that are in the form of a view set for indistinguishability. A *view set* is a pair (V, v), where V is a list of selection-projection queries $(q_1, q_2, ..., q_n)$ on Tbl, and v is a list of relations $(r_1, r_2, ..., r_n)$ that are the results, with duplicates preserved, of the corresponding queries. We may abbreviate (V, v) to v if V is understood. In this paper, "view", "query" and "query result" are used interchangeably when no confusion arises. *Note all query results preserve duplicates, hence, are multisets and all relational operations in this paper are on multisets.*

3.1 Checking for k-SIND

In this subsection, we will show that checking for k-SIND is intractable. Then, we will present a subcase where checking is tractable, before which the basic checking mechanism is presented. Finally, we will also present a conservative checking methods that always catch k-SIND violation, but may make mistakes when a view set actually provides k-SIND.

Complexity Checking for k-SIND is intractable. This is mainly because it is intractable to know whether a private value can associate with a particular B tuple by just looking at the view set.

Theorem 1. *Given a view set v containing only selection and projection, it is NP-hard to decide whether there exists an instance $I \in \mathcal{I}^v$ such that a tuple (b, p) is in I.*

The proof of the above theorem is by showing a reduction to our problem from the following NP-hard *Complement Boolean Auditing Problem* (whose complement, *Boolean auditing problem*, has been shown as coNP-hard [17]).

Theorem 2. *Given a view set v, whether v provides k-SIND is coNP-hard.*

We reduce the complement of the problem in Theorem 1 (that is, determining if a tuple (b, p) appears in at least one instance in $\mathcal{I}^v$) to the problem of checking k-SIND. Given any table Tbl and view set v, we construct another table Tbl' and view set v', such that v' violates 2-SIND iff (b, p) appears in at least one instance in $\mathcal{I}^v$. Because it is NP-hard to determine the latter by Theorem 1, it is coNP-hard to determine if v' satisfies 2-SIND.

Basic mechanism for checking First, we introduce an important property of SIND in Proposition 4. This property will be used in the subsequent checking methods.

Proposition 4. *Given a view set v and two tuples b_1 and b_2 in $\Pi_B(Tbl)$, b_1 and b_2 are* SIND *w.r.t. v, iff for each pair of P values p_1 and p_2 associated with b_1 and b_2, respectively, in an instance in $\mathcal{I}^v$, and each query q in v, we have $q(\{(b_1, p_1), (b_2, p_2)\}) = q(\{(b_1, p_2), (b_2, p_1)\})$*

Assume b_1 and b_2 are SIND. Then for each view q in v, $q(\{(b_1, p_1), (b_2, p_2)\} \cup I_o) = q(\{(b_1, p_2), (b_2, p_1)\} \cup I_o)$ (all query results are multisets and relational operations are multiset operations), where I_o is an instance such that $\{(b_1, p_1), (b_2, p_2)\} \cup I_o \in \mathcal{I}^v$. Since q only contains selection and projection, $q(\{(b_1, p_1), (b_2, p_2)\} \cup I_o) = q(\{(b_1, p_1), (b_2, p_2)\} \cup q(I_o)$ and $q(\{(b_1, p_2), (b_2, p_1)\} \cup I_o) = q(\{(b_1, p_2), (b_2, p_1)\} \cup q(I_o)$. Thus, we have $q(\{(b_1, p_1), (b_2, p_2)\}) = q(\{(b_1, p_2), (b_2, p_1)\})$. The other direction holds for the same reason.

We call the equation in this proposition *swap equation*. This proposition suggests SIND for selection-projection views has the property of being "local". Indeed, to check SIND between given two B tuples, we do not need to see other B tuples.

More specifically, this proposition says that given v and two SIND B tuples b_1 and b_2, for each query q in v, if the tuples (b_1, p_1) and (b_2, p_2) are in an instance that yields v, and we swap the private values of (b_1, p_1) and (b_2, p_2) to get the two new tuples, i.e., (b_1, p_2) and (b_2, p_1), then we know that $\{(b_1, p_2), (b_2, p_1)\}$ yields the same result of q as $\{(b_1, p_2), (b_2, p_1)\}$ does. This is a necessary and sufficient condition.

As a simple example, given two B tuples b_1 and b_2, if in all the instances in $\mathcal{I}^v$, we know they associate either with p_1 and p_2, respectively, or p_2 and p_3, respectively. Then b_1 and b_2 are SIND iff

$$q\begin{pmatrix}(b_1, p_1)\\(b_2, p_2)\end{pmatrix} = q\begin{pmatrix}(b_1, p_2)\\(b_2, p_1)\end{pmatrix} \;\&\; q\begin{pmatrix}(b_1, p_2)\\(b_2, p_3)\end{pmatrix} = q\begin{pmatrix}(b_1, p_3)\\(b_2, p_2)\end{pmatrix}$$

To satisfy swap equation

$$q\begin{pmatrix}(b_1, p_1)\\(b_2, p_2)\end{pmatrix} = q\begin{pmatrix}(b_1, p_2)\\(b_2, p_1)\end{pmatrix}$$

there are only two possibilities: one is

$$q((b_1, p_1)) = q((b_1, p_2)) \;\&\; q((b_2, p_2)) = q((b_2, p_1))$$

and the other is

$$q((b_1, p_1)) = q((b_2, p_1)) \;\&\; q((b_1, p_2)) = q((b_2, p_2))$$

If a view has a projection on P and p_1 is distinct from p_2, we can easily prove that we only need to check the latter condition. Moreover, *if the projection of q contains P, and b_1 and b_2 have at least two possible private values, then it is*

a necessary and sufficient condition for b_1 and b_2 being SIND that $q((b_1, p)) = q((b_2, p))$ holds for each possible value p.

For example, consider the view q in Figure 2 with the projection on P. Clearly, $q((t_9[B], H)) = q((t_{10}[B], H))$ and $q((t_9[B], C)) = q((t_{10}[B], C))$, where $H = Hypertension$ and $C = ChestPain$. Since H and C are the only possible values by looking at the view, we know $t_9[B]$ and $t_{10}[B]$ are SIND.

In the rest of this subsection, we use Proposition 4 to decide whether a view set v provides k-SIND.

Selection only on B attributes This subcase is common, especially in statistical databases, and hence is extensively studied with uncertainty measures[1, 17, 16]. In this subcase, each query in the view set has a selection condition only on the attributes in B. If so, checking for k-SIND can be done in polynomial time in the size of the private table and the number of views.

We assume the projection of each view contains P; otherwise, no private information is involved since the selection condition also does not have P and the view may be removed from our consideration. By Proposition 4, we have following conclusion for checking.

Proposition 5. *Given a view set v with selection conditions only on the attributes in B, two B tuples b_1 and b_2 are SIND if for each query $q = \Pi_X \sigma_C(Tbl)$ in v, we have $\Pi_{X-\{P\}}\sigma_C(b_1) = \Pi_{X-\{P\}}\sigma_C(b_2)$. The inverse ("only if") holds if b_1 and b_2 have at least two distinct possible private values.*

For each query q in v, if we have $\Pi_{X-\{P\}}\sigma_C(b_1) = \Pi_{X-\{P\}}\sigma_C(b_2)$, then $q(b_1, p) = q(b_2, p)$ holds for each p in the domain. Because C does not contain p, we can apply the conclusion from Proposition 4. Thus, each pair of possible tuples (b_1, p_1) and (b_2, p_2) satisfies swap equation. Otherwise, $q(b_1, p) = q(b_2, p)$ can not hold, hence, swap equation can not be satisfied if b_1 and b_2 have at least two distinct possible private values (if there is only one possible private value, the swap results in the same instance, hence swap equation must be satisfied).

We assume that the set of k indistinguishable B tuples must have at least two distinct possible private values; otherwise, it must not be safe. Thus, the condition of Proposition 5 is a necessary and sufficient condition. By this proposition, we can present an efficient checking method through partitioning. The basic idea is that for each view, we can partition tuples such that each set in the partition are SIND w.r.t. this view, and we then intersect these partitions.

As an example of this procedure, consider the two views

$$\Pi_{Race,Problem}\ \sigma_{Zip='22030'}\ (Tbl) \text{ and}$$
$$\Pi_{Gender,Problem}\ \sigma_{Race='White'}\ (Tbl).$$

We partition the B tuples as Figure 4 (a) by the first view and by the second view as Figure 4 (b); the final result in Figure 4 (c) is the intersection of the two partitions shown in (a) and (b). For each view, the selected tuples that have the same values on the projection are grouped in the same set of the partition, $(Zip, Race)$ for the first and $(Race, Gender)$ for the second; the tuples that are not selected are grouped into another set in the partition. If two B tuples are in

(22030, White)	$t_1[B]$, $t_2[B]$, $t_3[B]$
(22030, Black)	$t_4[B]$
Others not selected	$t_5[B]$, $t_6[B]$, ..., $t_{12}[B]$

(a) By the first view

(White, Male)	$t_1[B]$, $t_2[B]$, $t_3[B]$, $t_6[B]$, $t_8[B]$, $t_{11}[B]$, $t_{12}[B]$
(White, Female)	$t_6[B]$
Others not selected	$t_4[B]$, $t_5[B]$, $t_7[B]$, $t_9[B]$, $t_{10[B]}$

(a) By the second view

$t_1[B]$, $t_2[B]$, $t_3[B]$
$t_4[B]$
$t_5[B]$, $t_7[B]$, $t_9[B]$, $t_{10}[B]$
$t_6[B]$
$t_8[B]$, $t_{11}[B]$, $t_{12}[B]$

(c) The final partition

Fig. 4. The partition of B tuples by views

the same block of the final partition, they are SIND. In this case, we only have 1-SIND.

Now we analyze the computational complexity of this checking procedure. The partitioning for each view q in this procedure needs to search for the set of B tuples yielding the same result of q. Such searching can be done using a hash data structure, hence is constant time. And each partition needs to scan all the B tuples in Tbl once. Thus the computing time is $O(nS)$, where S is the size of Tbl and n is the number of views in the view set.

Conservative checking Because checking a view set for k-SIND is generally intractable, we may want to perform a conservative-style checking that is polynomial time and suitable for all cases. With a conservative algorithm, we always catch k-SIND violation, but we may make mistakes when a view set actually provides k-SIND.

First, we can use a conservative checking method for each single view q. The basic idea is that if two B tuples have the same characteristics in the selection condition and have the same value on the B attributes in the projection of q, then they are SIND for q.

More specifically, if two B tuples b_1 and b_2 have the same values on the B attributes in the projection of q, and after substitute B with b_1 and b_2, respectively, in the selection condition, the two substituted conditions have the same set of P values making the conditions true, then they are SIND. We can see that this method does not look for the possible private values. Thus, it has the

similar procedure as the checking method for the case where v selects only B attributes in Subsection 3.1. That is, generate a partition for each view by the corresponding attribute values of B tuples and intersect these partitions.

For example, consider the view $\Pi_{Zip}\sigma_{Charge>Salary}(Tbl)$ on the table Tbl. Each distinct $Charge$ value c has the different set of $Salary$ values making the selection condition true when you substitute $Charge$ with c in the condition. Thus, if two B tuples have the same $(Zip, Charge)$ value, then we take them as SIND; otherwise, we do not.

Clearly, this method for checking each single view is polynomial time. Then we can use the following conclusion to check a set of views. We have that if two b tuples are SIND w.r.t. each view in v, then then b_1 and b_2 are SIND w.r.t. v. Therefore, we can generate a SIND partition for each view in v, and then intersection these partitions to get a SIND partition for v. All the B tuples in the same set of the partition for v must be SIND. Then if the cardinality of each set of the final partition is at least k, the view set provides k-SIND. Otherwise, it may not do.

Checking in this way can be done in polynomial time, and is a conservative method to check the sufficient condition of SIND. We believe this conservative checking is practical, since we do not check what possible private values each individual has. In fact, it has the similar idea as k-anonymization methods [24, 25, 21, 14, 18]. Indeed, this checking does not look for the possible private values but looks at the public values while k-anonymization recodes only the public values of tuples to achieve k-anonymity.

3.2 Checking for k-RSIND

In this section, we turn to checking whether there exists an RSIND partition such that the cardinality of each set in the partition is at least k. By Proposition 3, we can check whether there exists this kind of partition by looking for the B tuples in each set that are RSIND from each other w.r.t. their current private values.

To do this, given a set T of tuples in Tbl, we need to check whether for each pair of B tuples b_1 and b_2 in $\Pi_B(T)$, each pair of P values p_1 and p_2 in $\Pi_P(T)$, and each instance I in $\mathcal{I}^v$ that contains (b_1, p_1) and (b_2, p_2), there exists an instance I' in $\mathcal{I}^v$ such that $I' = (I - \{(b_1, p_1), (b_2, p_2)\}) \cup \{(b_1, p_2), (b_2, p_1)\}$. Through the similar deduction as in Proposition 4, this swap is equivalent to that for each query q in v, we must have $q(\{(b_1, p_1), (b_2, p_2)\}) = q(\{(b_1, p_2), (b_2, p_1)\})$.

For each pair of B tuples in $\Pi_B(T)$ and each pair of private values in $\Pi_P(T)$, this swap equation needs to be checked. Then, for n tuples, it needs to be checked $O(n^4)$ times (there are $O(n^2)$ pairs of B tuples and $O(n^2)$ pairs of private values), where n is the cardinality of T. Obviously, this is costly.

However, in most cases, if each two B tuples in $\Pi_B(T)$ can swap their current values, then the two B tuples can swap each two private values in $\Pi_P(T)$. For instance, given $T = \{(b_1, p_1), \{(b_2, p_2), \{(b_3, p_3)\}$, if b_1 and b_2 can swap for p_1 and p_2, b_2 and b_3 for p_2 and p_3, and b_3 and b_1 for p_3 and p_1, then any two

B tuples, for example, b_1 and b_2, can swap for all pairs of the current private values, p_1 and p_2, p_2 and p_3, and p_3 and p_1.

For convenience, we introduce another concept. Given two B tuples b_1 and b_2, and (b_1, p_1) and (b_2, p_2) in the current table, we say b_1 and b_2 are *CSIND* (currently SIND) if for each query q in v, we have either (1) q contains the projection on P, and $q((b_1, p_1)) = q((b_2, p_1))$ and $q((b_1, p_2)) = q((b_2, p_2))$, or (2) q does not contain the projection on P, but $q((b_1, p_1)) = q((b_1, p_2))$ and $q((b_2, p_1))) = q((b_2, p_2))$. Intuitively, if b_1 and b_2 are CSIND, we can swap their private values in the current table without affecting the view set.

Clearly, if each two tuples in $\Pi_B(T)$ are CSIND, then each two B tuples in $\Pi_B(T)$ satisfies swap equation for all the private values in $\Pi_P(T)$, hence, $\Pi_B(T)$ are RSIND from each other. Indeed, if each q contains the projection on P, this is a necessary and sufficient condition; otherwise, it is a sufficient condition.

Therefore, we apply the following checking method. If we can find a maximal partition over the B tuples $\Pi_B(Tbl)$ such that each pair of B tuples in each set in the partition are CSIND, then this partition is an RSIND partition. Here, "maximal" means that the union of any two sets in the partition cannot result in a set in which each pair of B tuples are still CSIND. In this way, we can find an RSIND partition by checking whether each pair of tuples in the current table is able to swap their private values. This provides a conservative checking algorithm for k-RSIND as follows.

Construct a graph G. Each tuple maps to a node. If two tuples are CSIND, which can be easily checked based on the current private table, add an edge between the corresponding nodes. Then a complete subgraph of G is a subset of an RSIND set. Therefore, finding an RSIND partition can be transformed to finding a maximal clique partition. If each query of v contains the projection on P, the above checking algorithm is a precise (not conservative) algorithm.

It is known, however, that finding a clique partition with each block's size of at least k is NP-hard [13]. It is not difficult to prove that it is NP-hard to decide whether v provides k-RSIND in the special case where each query of v contains the projection on P. Therefore, given a released view set v, it is NP-hard to decide whether v provides k-RSIND.

Nevertheless, we can use the heuristic algorithms in [13] to find a clique partition with each block size at least k. This will result in a conservative algorithm even for the special case where each query in v contains the projection on P.

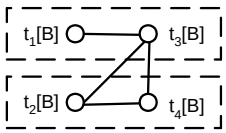

Fig. 5. An RSIND partition for the views in Figure 3 maps to a maximal clique partition.

For example, consider the views in Figure 3. We construct a graph as Figure 5. Each edge represents that the B tuples corresponding to the two adjacent nodes are CSIND. An RSIND partition maps to a maximal clique partition in the graph.

4 Related Work

The most relevant work to indistinguishability is k-anonymization, which focuses on how to gain k-anonymity by recoding the quasi-IDs in a single view, e.g., [24, 25, 21, 14, 18]. Recently, there is another work [19] aiming to achieve good uncertainty while gaining k-anonymity by imposing additional requirements on anonymization. But k-anonymity applies only to the case where released data is an anonymized table. In our work, we introduced different definitions of indistinguishability that apply to more general situations, and focused on checking data base views against these indistinguishability metrics. We also discussed the relationship between k-anonymity and indistinguishability.

Some work studies the privacy or secrecy disclosure by general database views. The conditions of perfect secrecy are studied in [22, 8] using probability model, and in [28] using query conditional containment. In this paper, we addressed the case where we intend to release data if some partial disclosure by database views is tolerated, and hence the disclosure needs to be measured.

Except for the study of k-anonymity, the privacy metrics used in prior work have mainly been based on uncertainty of private property values, i.e., the uncertainty what private value an individual has. These metrics can be classified into two categories: non-probabilistic and probabilistic.

The non-probabilistic metrics are mainly used in the fields of inference problem of statistical databases [1, 17, 16, 26], multilevel databases [20, 5] and general purpose databases [7, 5, 27]. The most often used one is that if the private value of an individual cannot be uniquely inferred, released data about the individual are considered safe [1, 20, 7, 17, 5, 16]. The other one is the cardinality of the set of possible private values for each individual, among which attackers cannot determine which one is the actual one [26, 27] (The metric used in [27] is an uncertainty metric in spite of the notion of k-anonymity introduced).

In the above fields, some uncertainty metrics are based on probability. Authors use the probability value associated with the actual value [12, 16] or the variance of the probability distribution of private values [1, 23]. Most work in privacy-preserving data mining uses probability-based metrics. Their metrics are based on only the characteristics of the *a posteriori* probability distribution of private values [3, 2, 10], or on both *a priori* and the *a posteriori* distribution [2, 9, 15, 4]. The work in [6] uses indistinguishability based on probability "distance" as privacy metric.

In this work, we used symmetry-based indistinguishability metrics. And we illustrated that uncertainty needs to be supplemented with indistinguishability.

5 Conclusions

In this paper, we identified a requirement of privacy in data release, namely indistinguishability, in addition to uncertainty. We first gave two definitions of indistinguishability, namely, SIND and RSIND. Then we focused on checking database views against these indistinguishability metrics. Generally, checking for k-SIND is intractable. We presented a case where polynomial algorithms are possible. Furthermore, we presented a conservative checking method. Checking for RSIND is easy, but checking for k-RSIND is intractable and can be done in a conservative way with heuristic polynomial algorithms.

References

1. N. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4):515–556, December 1989.
2. D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, 2001.
3. R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD Conference)*, pages 439–450, 2000.
4. S. Agrawal and J. R. Haritsa. A framework for high-accuracy privacy-preserving mining. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 193–204, 2005.
5. A. Brodsky, C. Farkas, and S. Jajodia. Secure databases: Constraints, inference channels, and monitoring disclosures. *IEEE Transactions on Knowledge and Data Engineering*, 12(6):900–919, 2000.
6. S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography, Second Theory of Cryptography Conference (TCC)*, pages 363–385, 2005.
7. H. S. Delugach and T. H. Hinke. Wizard: A database inference analysis and detection system. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):56–66, 1996.
8. A. Deutsch and Y. Papakonstantinou. Privacy in database publishing. In *Database Theory - ICDT 2005, 10th International Conference*, pages 230–245, 2005.
9. A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 211–222, 2003.
10. A. V. Evfimievski, R. Srikant, R. Agrawal, and J. Gehrke. Privacy preserving mining of association rules. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 217–228, 2002.
11. R. Gavison. Privacy and the limits of the law. In D. G. Johnson and H. Nissenbaum, editors, *Computers, Ethics, and Social Values*. 1995.
12. J. Hale and S. Shenoi. Catalytic inference analysis: Detecting inference threats due to knowledge discovery. In *Proceedings of the 1997 IEEE Symposium on Security and Privacy*, pages 188–199, 1997.

13. X. Ji and J. E. Mitchell. Branch-and-price-and-cut on clique partition problem with minimum clique size requirement. In *IMA Special Workshop: Mixed-Integer Programming*, 2005.
14. R. J. B. Jr. and R. Agrawal. Data privacy through optimal k-anonymization. In *Proceedings of the 21st International Conference on Data Engineering (ICDE)*, pages 217–228, 2005.
15. M. Kantarcioglu, J. Jin, and C. Clifton. When do data mining results violate privacy? In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 599–604, 2004.
16. K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 118–127, 2005.
17. J. M. Kleinberg, C. H. Papadimitriou, and P. Raghavan. Auditing boolean attributes. In *Proceedings of the Nineteenth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pages 86–91, 2000.
18. K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: Efficient full-domain k-anonymity. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD Conference)*, pages 49–60, 2005.
19. A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *Proceedings of the 22nd International Conference on Data Engineering (ICDE)*, pages 24–35, 2006.
20. D. G. Marks. Inference in MLS database systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(1):46–55, 1996.
21. A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proceedings of the Twenty-third ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 223–228, 2004.
22. G. Miklau and D. Suciu. A formal analysis of information disclosure in data exchange. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD Conference)*, pages 575–586, 2004.
23. K. Muralidhar and R. Sarathy. Security of random data perturbation methods. *ACM Transactions on Database Systems (TODS)*, 24(4):487–493, 1999.
24. P. Samarati. Protecting respondents' identities in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.
25. L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):571–578, 2002.
26. L. Wang, D. Wijesekera, and S. Jajodia. Cardinality-based inference control in sum-only data cubes. In *Proceedings of 7th European Symposium on Research in Computer Security (ESORICS)*, pages 55–71, 2002.
27. C. Yao, X. S. Wang, and S. Jajodia. Checking for k-anonymity violation by views. In *Proceedings of the 31st International Conference on Very Large Data Bases (VLDB)*, pages 910–921, 2005.
28. Z. Zhang and A. O. Mendelzon. Authorization views and conditional query containment. In *Database Theory - ICDT 2005, 10th International Conference*, pages 259–273, 2005.