Approximation Algorithms for k-Anonymity[*]

Gagan Aggarwal GAGAN@CS.STANFORD.EDU
Google Inc., Mountain View, CA

Tomas Feder TOMAS@THEORY.STANFORD.EDU
268 Waverley St., Palo Alto, CA

Krishnaram Kenthapadi KNGK@CS.STANFORD.EDU
Stanford University, Stanford, CA

Rajeev Motwani RAJEEV@CS.STANFORD.EDU
Stanford University, Stanford, CA

Rina Panigrahy RINAP@CS.STANFORD.EDU
Stanford University, Stanford, CA

Dilys Thomas DILYS@CS.STANFORD.EDU
Stanford University, Stanford, CA

An Zhu ANZHU@CS.STANFORD.EDU
Google Inc., Mountain View, CA

Abstract

We consider the problem of releasing a table containing personal records, while ensuring individual privacy and maintaining data integrity to the extent possible. One of the techniques proposed in the literature is k*-anonymization*. A release is considered k*-anonymous* if the information corresponding to any individual in the release cannot be distinguished from that of at least $k-1$ other individuals whose information also appears in the release. In order to achieve k*-anonymization*, some of the entries of the table are either suppressed or *generalized* (e.g. an Age value of 23 could be changed to the Age range 20-25). The goal is to lose as little information as possible while ensuring that the release is k*-anonymous*. This optimization problem is referred to as the k*-Anonymity problem*. We show that the k-Anonymity problem is NP-hard even when the attribute values are ternary and we are allowed only to suppress entries. On the positive side, we provide an $O(k)$-approximation algorithm for the problem. We also give improved positive results for the interesting cases with specific values of k — in particular, we give a 1.5-approximation algorithm for the special case of 2-Anonymity, and a 2-approximation algorithm for 3-Anonymity.

[*]. A preliminary version of this paper appeared in the *Proceedings of the 10th International Conference on Database Theory (ICDT'05)* (Aggarwal, Fèder, Kenthapadi, Motwani, Panigrahy, Thomas, and Zhu, 2005).

Manuscript received 6 Aug 2005, accepted 8 Nov 2005, published 20 Nov 2005

Keywords: Anonymity, Approximation Algorithms, Database Privacy

1. Introduction

There has been a tremendous growth in the amount of personal data that can be collected and analyzed. Data mining tools are increasingly being used to infer trends and patterns. In many scenarios, access to large amounts of *personal data* is essential in order for accurate inferences to be drawn. For example, at the beginning of an epidemic, a single hospital might see only a few isolated cases, whereas the combined patient pool of a group of hospitals might be sufficient to infer the outbreak of an epidemic. However, the use of data containing personal information has to be restricted in order to protect individual privacy.

One possible solution is that instead of releasing the entire database, the database owner answers aggregate queries posed by medical researchers after ensuring that answers to the queries do not reveal sensitive information. In *query auditing* (Kleinberg, Papadimitriou, and Raghavan, 2003; Dinur and Nissim, 2003; Kenthapadi, Mishra, and Nissim, 2005), a query is denied if the response could reveal sensitive information and answered otherwise. On the other hand, in *output perturbation* (Dinur and Nissim, 2003; Dwork and Nissim, 2004; Blum, Dwork, McSherry, and Nissim, 2005), the database owner provides a perturbed answer to each query. These methods require the researchers to formulate their queries without access to any data. In this case, one can also use techniques from *secure multi-party computation* (Yao, 1986; Goldreich, Micali, and Wigderson, 1987; Lindell and Pinkas, 2002; Aggarwal, Mishra, and Pinkas, 2004; Freedman, Nissim, and Pinkas, 2004). However, many of the data-mining tasks are inherently ad hoc and the data mining researchers need to examine the data in order to discover data aggregation queries of interest. In such cases, query auditing, output perturbation and secure function evaluation techniques do not provide an adequate solution, and we need to release an anonymized view of the database that enables the computation of non-sensitive query aggregates, perhaps with some error or uncertainty.

One approach to anonymization uses *data sanitization* techniques in order to hide the exact values of the data (Agrawal and Srikant, 2000; Agrawal and Aggarwal, 2001; Evfimievski, Gehrke, and Srikant, 2003; Agrawal, Srikant, and Thomas, 2005; Chawla, Dwork, McSherry, Smith, and Wee, 2005). However, this may not be suitable if one wants to draw inferences with 100% confidence. Another approach is to *suppress* some of the data values, while releasing the remaining data values exactly. We note that suppressing just the identifying attributes is not sufficient to protect privacy. For example, consider the following table which is part of a medical database, with the identifying attributes such as name and social security number removed.

Age	Race	Gender	Zip Code	Diseases
47	White	Male	21004	Common Cold
35	White	Female	21004	Flu
27	Hispanic	Female	92010	Flu
27	White	Female	92010	Hypertension

By joining this table with public databases (such as a voter list), non-identifying attributes, such as Age, Race, Gender and Zip Code in the above table, can together be used to identify individuals. In fact, Sweeney (2000) observed that for 87% of the population in the United States, the combination of Date of Birth, Gender and Zip Code corresponded to a unique person.

In order to ensure the protection of privacy, we adopt the k-Anonymity model that was proposed by Samarati and Sweeney (Samarati and Sweeney, 1998; Samarati, 2001; Sweeney, 2002). Suppose we have a table consisting of n tuples each having m quasi-identifying attributes (Age, Race, Gender and Zip Code in the above table), and let $k > 1$ be an integer. The k-Anonymity framework provides for generalization of entries (generalization entails replacing an entry value with a less specific but semantically consistent value; a more formal description can be found in Section 2) in addition to suppression. The idea is to suppress/generalize some of the entries in the table so as to ensure that *for each tuple in the modified table, there are at least* $k-1$ *other tuples in the modified table that are identical to it along the quasi-identifying attributes.* The objective is to minimize the extent of suppression and generalization. Note that entries in the column corresponding to the sensitive attribute (“Diseases” in the above example) are not altered. The following is an example of a k-anonymized table for $k = 2$.

Age	Race	Gender	Zip Code	Diseases
*	White	*	21004	Common Cold
*	White	*	21004	Flu
27	*	Female	92010	Flu
27	*	Female	92010	Hypertension

A k-anonymized table protects individual privacy in the sense that, even if an adversary has access to all the quasi-identifying attributes of all the individuals represented in the table, he would not be able to track down an individual’s record further than a set of at least k records, in the worst case. Thus, releasing a table after k-anonymization prevents definitive *record linkages* with publicly available databases, and keeps each individual hidden in a crowd of $k-1$ other people. The privacy parameter k must be chosen according to the application in order to ensure the required level of privacy.

2. Model and Results

We now formally define the problem of k-Anonymity and state our results. The input is a table having n rows each with m quasi-identifying attributes. We view the table as consisting of n m-dimensional vectors: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \Sigma^m$.

We first define a special case of the problem called *k-Anonymity with Suppression*, where suppression is the only permitted operation. A *k-Anonymous suppression function* t maps each $\mathbf{x}_i$ to $\tilde{\mathbf{x}}_i$ by replacing some components of $\mathbf{x}_i$ by $*$ (which corresponds to hiding those components of $\mathbf{x}_i$), so that every $\tilde{\mathbf{x}}_i$ is identical to at least $k-1$ other $\tilde{\mathbf{x}}_j$s. This results in a partition of the n row vectors into *clusters* of size at least k each. The cost of the suppression, $c(t)$ is the total number of hidden entries, or equivalently, the total number of $*$s in all the $\tilde{\mathbf{x}}_i$s.

k-Anonymity with Suppression: *Given* $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$*, and an Anonymity parameter* k*, obtain a* k*-Anonymous suppression function* t *so that* $c(t)$ *is minimized.*

Next, we define the problem of *k-Anonymity with Generalization*, where in addition to suppressing entry values, we are also allowed to replace them with less specific but semantically consistent values. For example, we can make a date less specific by omitting the day and revealing just the month and year. We assume that for each attribute, a generalization hierarchy is specified as part of the input (Samarati and Sweeney, 1998; Samarati, 2001). For an attribute, each level of generalization corresponds to a partition of the attribute domain. A partition corresponding to any given level of the generalization hierarchy is a refinement of the partition corresponding to the next higher level. Singleton sets correspond to absence of generalization, while the partition consisting of a single set containing the whole domain corresponds to the highest level of generalization. Consider the example shown in Figure 2. The attribute "Quality" has a domain consisting of values $A+$, A, $A-$, $B+$, B and $B-$ and has two levels of generalization. In the absence of generalization, the value of this attribute is reported exactly. The first level of generalization corresponds to the partition $\{\{A+, A, A-\}, \{B+, B, B-\}\}$. In order to generalize an entry with value "A" to the first level of generalization, it is replaced with the set $\{A+, A, A-\}$. The next higher level of generalization (also the highest level in this case) corresponds to replacing the entry with the set containing the whole domain, which is equivalent to suppressing the entry.

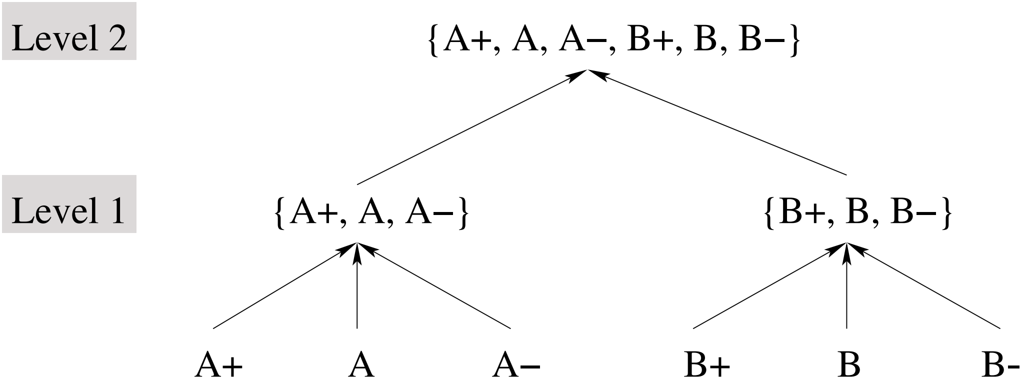


Figure 1: A possible generalization hierarchy for the attribute "Quality".

Let the j^{th} attribute have domain D^j and l_j levels of generalization. Let the partition corresponding to the h^{th} level of generalization be D^j_h for $1 \leq h \leq l_j$, with $D^j_0 = D^j$. Let a value $y \in D^j$ when generalized to the h^{th} level be denoted by $g_h(y)$, e.g., $g_1(A) = \{A+, A, A-\}$. A *generalization function* h is a function that maps a pair (i, j), $i \leq n$, $j \leq m$ to a level of generalization $h(i, j) \leq l_j$. Semantically, $h(i, j)$ denotes the level to which j^{th} component of the i^{th} vector (or the $(i, j)^{th}$ entry in the table) is generalized. Let $h(\mathbf{x}_i)$ denote the *generalized* vector corresponding to $\mathbf{x}_i$, i.e. $h(\mathbf{x}_i) = (g_{h(i,1)}(x_i[1]), g_{h(i,2)}(x_i[2]) \ldots, g_{h(i,m)}(x_i[m]))$. A generalization function is said to be k-Anonymous if for every i, $h(\mathbf{x}_i)$ is identical to $h(\mathbf{x}_j)$ for at least $k-1$ values of $j \neq i$.

Consider a k-Anonymous generalization function h. It incurs a cost of r/l_j whenever it generalizes a value for the j^{th} attribute to the r^{th} level. The total cost incurred by the generalization function h is defined as the sum of the costs incurred over all the entries of the table, i.e. cost$(h) = \sum_i \sum_j h(i, j)/l_j$. Now we are ready to give a formal definition of the problem.

k-Anonymity with Generalization: *Given $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in \Sigma^m$, and an Anonymity parameter k, obtain a k-Anonymous generalization function h such that cost(h) is minimized.*

Note that the problem of k-Anonymity with Suppression is a special case of the problem of k-Anonymity with Generalization, with only one level of generalization (corresponding to hiding the entry completely) for every attribute.

Clearly the decision version of both of these problems is in NP, since we can verify in polynomial time if the solution is k-Anonymous and the suppression cost less than a given value. We show that k-Anonymity with Suppression is NP-hard even when the alphabet size $|\Sigma| = 3$. Note that this automatically implies NP-hardness of k-Anonymity with Generalization. This improves upon the NP-hardness result of Meyerson and Williams (2004) which required an alphabet size of n. On the positive side, we provide an $O(k)$-approximation algorithm for k-Anonymity with Generalization for arbitrary k and arbitrary alphabet size, using a graph representation. This improves upon the previous best-known approximation guarantee of $O(k \log k)$ for k-Anonymity with Suppression (Meyerson and Williams, 2004). We also show that it is not possible to achieve an approximation factor better than $\Theta(k)$ using the graph representation approach. For a binary alphabet, we provide improved approximation algorithms for $k = 2$ (an approximation factor of 1.5) and $k = 3$ (an approximation factor of 2).

The rest of the paper is organized as follows. We establish the NP-hardness of k-Anonymity with Suppression in Section 3. We then present an $O(k)$-approximation algorithm for k-Anonymity with Generalization in Section 4. Next, in Sections 5 and 6, we provide a 1.5 approximation algorithm for the 2-Anonymity problem with binary alphabet, and a 2-approximation algorithm for 3-Anonymity with binary alphabet. Finally, we conclude with some future research directions in Section 7.

3. NP-hardness of k-Anonymity with Suppression

Theorem 1 *k-Anonymity with Suppression is NP-hard even for a ternary alphabet, i.e., $(\Sigma = \{0, 1, 2\})$.*

Proof In this proof, k-Anonymity refers to the problem of k-Anonymity with Suppression. We give a reduction from the NP-hard problem of EDGE PARTITION INTO TRIANGLES (Kann, 1994) which is defined as follows: *Given a graph $G = (V, E)$ with $|E| = 3m$ for some integer m, can the edges of G be partitioned into m edge-disjoint triangles?*

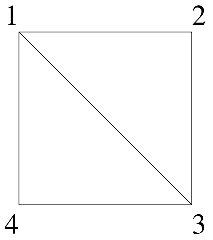
Given an instance of the above problem, $G = (V, E)$ with $3m$ edges (since the above problem is NP-hard even for simple graphs, we will assume that the graph G is simple), we create a preliminary table T with $3m$ rows — one row for each edge. For each of the n vertices of G, we create an attribute (column). The row corresponding to edge (a, b), referred to as r_{ab}, has ones in the positions corresponding to a and b and zeros everywhere else. Let a star with four vertices (having one vertex of degree 3) be referred to as a 4-star.

Equivalence to edge partition into triangles and 4-stars. We first show that the cost of the optimal 3-Anonymity solution for the table T is at most $9m$ if and only if E can be partitioned into a collection of m disjoint triangles and 4-stars. First suppose that such a partition of edges is given.

Consider any triangle (with a, b, c as its vertices). By suppressing the positions a, b and c in the rows r_{ab}, r_{bc} and r_{ca}, we get a cluster containing three rows, with three $*$s in each modified row. Now consider a 4-star with vertices a, b, c, d, where d is the center vertex. By suppressing the positions a, b and c in the rows r_{ad}, r_{bd} and r_{cd}, we get a cluster containing three rows with three $*$s in each modified row. Thus we obtain a solution to 3-Anonymity of cost $9m$.

On the other hand, suppose that there is a 3-Anonymity solution of cost at most $9m$. Since G is simple, any three rows are distinct and differ in at least 3 positions. Hence there should be at least three $*$s in each modified row, so that the cost of the solution is at least $9m$. This implies that the solution cost is exactly $9m$ and each modified row has exactly three $*$s. Since any cluster of size more than three will have at least four $*$s in each modified row, it follows that each cluster has exactly three rows. There are exactly two possibilities: the corresponding edges form either a triangle or a 4-star, and each modified row in a triangle has three $*$s and zeros elsewhere while each modified row in a 4-star has three $*$s, single 1 and zeros elsewhere. Thus, the solution corresponds to a partition of the edges of the graph into triangles and 4-stars.

Equivalence to edge partition into triangles. Since we want a reduction from EDGE PARTITION INTO TRIANGLES, we create a table T' by "replicating" the columns of T so as to force the 4-stars to pay more $*$s. Let $t = \lceil \log_2(3m + 1) \rceil$. In the new table T', every row has t *blocks*, each of which has n columns. Consider an arbitrary ordering of the edges in E and express the rank of an edge $e = (a, b)$, in this ordering, in binary notation as $e_1 e_2 \dots e_t$. In the row corresponding to edge e, each block has zeros in all positions except a and b. A block can be in one of two configurations: $conf_0$ has a 1 in position a and a 2 in position b while $conf_1$ has a 2 in position a and a 1 in position b. The i^{th} block in the row corresponding to e has configuration $conf_{e_i}$. For example, consider the graph shown in Figure 3. Suppose the edges $(3, 4), (1, 4), (1, 2), (1, 3), (2, 3)$ are ranked 1 (i.e. $(001)_2$) through 5 (i.e. $(101)_2$) respectively. Then, the table in Figure 3 represents the 3-Anonymity instance corresponding to the graph, with the i^{th} row in the table representing the vector corresponding to the edge ranked i.



(3,4)	0	0	1	2	0	0	1	2	0	0	2	1
(1,4)	1	0	0	2	2	0	0	1	1	0	0	2
(1,2)	1	2	0	0	2	1	0	0	2	1	0	0
(1,3)	2	0	1	0	1	0	2	0	1	0	2	0
(2,3)	0	2	1	0	0	1	2	0	0	2	1	0

Figure 2: The table shows the 3-anonymity instance corresponding to the graph on the left when the edges $(3, 4), (1, 4), (1, 2), (1, 3), (2, 3)$ are ranked 1 through 5 respectively.

We will now show that the cost of the optimal 3-Anonymity solution on T' is at most $9mt$ if and only if E can be partitioned into m disjoint triangles.

Suppose that E can be partitioned into m disjoint triangles. As earlier, every triangle in such a partition corresponds to a cluster with $3t$ $*$s in each modified row. Thus we get a 3-Anonymity solution of cost $9mt$.

For the converse, suppose that we are given a 3-Anonymity solution of cost at most $9mt$. Again, any three rows differ in at least $3t$ positions so that the cost of any solution is at least $9mt$. Hence the solution cost is exactly $9mt$ and each modified row has exactly $3t$ $*$s. Thus, each cluster has exactly three rows. We claim that the corresponding edges should form a triangle. We can see this as follows: suppose to the contrary the three rows form a 4-star. Let the common vertex be v. Consider the ternary digit $\in \{1, 2\}$ assigned by each of the three edges to v in $conf_0$ — two of the three edges must have assigned the same digit to v. Since these two edges differ in rank, they must have a different configuration (and therefore, a different digit in the column corresponding to v) in at least one of the blocks. Thus, the rows corresponding to the three edges contain an additional $*$ corresponding to vertex v in addition to the $3t$ $*$s corresponding to the remaining three vertices, contradicting the fact that each row has exactly $3t$ $*$s. ■

The above proof shows that k-Anonymity is NP-hard even with a ternary alphabet for $k = 3$. By reduction from EDGE PARTITION INTO r-CLIQUES (Kann, 1994), we can extend the above proof for $k = \binom{r}{2}$, for $r \geq 3$. By replicating the graph in the above reduction, we can further extend the proof for $k = \alpha\binom{r}{2}$ for any integer α and $r \geq 3$.

4. Algorithm for General k-Anonymity

In this section, we study the problem of k-Anonymity with Generalization for general k and arbitrary alphabet size, and give an $O(k)$-approximation algorithm for the problem. In this section, k-Anonymity refers to the problem of k-Anonymity with Generalization.

Construction of Graph. Given an instance of the k-Anonymity problem, we create an edge-weighted complete graph $G = (V, E)$. The vertex set V contains a vertex corresponding to each vector in the k-Anonymity problem. For two rows $\mathbf{a}$ and $\mathbf{b}$, let the unscaled generalization cost for the j^{th} component, $h_{a,b}(j)$, refer to the lowest level of generalization for attribute j for which the j^{th} components of both $\mathbf{a}$ and $\mathbf{b}$ are in the same partition, i.e. the lowest level for which both have the same generalized value. The weight, $w(e)$, of an edge $e = (a, b)$ is the sum over all components j of the scaled generalization cost, i.e. $w(e) = \sum_j h_{a,b}(j)/l_j$ (recall that the scaling factor l_j corresponds to the total number of levels of generalizations for the j^{th} attribute). The j^{th} attribute is said to contribute a weight of $h_{a,b}(j)/l_j$ to the edge e.

Limitations of the Graph Representation. As mentioned in Section 2, with this representation, we lose some information about the structure of the problem, and cannot achieve a better than $\Theta(k)$ approximation factor for the k-Anonymity problem. We show this by giving two instances (on binary alphabet) whose k-Anonymity cost differs by a factor of $\Theta(k)$, but the corresponding graphs for both the instances are identical. Let $l = 2^{k-2}$. For the first instance, take k vectors with kl-dimensions each. The bit positions $(i-1)l + 1$ to il are referred to as the i^{th} block of a vector. The i^{th} vector has ones in the i^{th} block and zeros everywhere else. The k-Anonymity cost for this instance is $k^2 l$. For the second instance, take k vectors with $4l = 2^k$ dimensions each. The i^{th} vector breaks up its 2^k dimensions into 2^i equal-sized blocks and has ones in the odd blocks and zeros in the even blocks. This instance incurs a k-Anonymity cost of $4kl$. Note that the graph corresponding to both the instances is a k-clique with all the pairwise distances being $2l = 2^{k-1}$.

Definition 2 (Charge of a vertex) *For any given k-Anonymity solution, define the* charge *of a vertex to be the total generalization cost of the vector it represents.*

Idea Behind the Algorithm. Let OPT denote the cost of an optimal k-Anonymity solution, i.e., OPT is the sum of the charges of all the vertices in an optimal k-Anonymity solution. Let $F = \{T_1, T_2, \ldots, T_s\}$, a spanning forest (i.e. a forest containing all the vertices) in which each tree T_i has at least k vertices, be a subgraph of G. This forest describes a feasible partition for the k-Anonymity problem. In the k-Anonymity solution as per this partition, the charge of each vertex is no more than the weight of the tree containing the vertex; recall that the weight of a tree T_i is given by $W(T_i) = \Sigma_{e \in E(T_i)} w(e)$, where $E(T_i)$ denotes the set of edges in tree T_i. We can see this as follows: if attribute j has to be generalized to level r for the vertices in tree T_i (note that an attribute is generalized to the same level for all rows in a cluster), there must exist a pair of vertices (a, b) in the cluster which have an unscaled generalization cost $h_{a,b}(j)$ equal to r. Thus, attribute j contributes a weight of at least r/l_j to the length of all paths (in G) between a and b. In particular, attribute j contributes a weight of at least r/l_j to the weight of tree T_i. Next, we sum the charges of all the vertices to get that the k-Anonymity cost of the partition corresponding to the forest F is at most $\Sigma_i |V(T_i)| W(T_i)$. We will refer to this as the k-Anonymity cost of the forest. Note that the weight of a forest is simply the sum of the weights of its trees. Hence, the ratio of the k-Anonymity cost to the weight of a forest is at most the number of vertices in the largest tree in the forest. This implies that if we can find a forest with the size of the largest component at most L and weight at most OPT, then we have an L-approximation algorithm. Next, we present an algorithm that finds such a forest with $L \leq \max\{2k-1, 3k-5\}$.

The algorithm has the following overall structure, which is explained in more detail in the next two subsections.

Outline of the Algorithm:

1. Create a forest G with cost at most OPT. The number of vertices in each tree is at least k.
2. Compute a decomposition of this forest (deleting edges is allowed) such that each component has between k and $\max\{2k-1, 3k-5\}$ vertices. The decomposition is done in a way that does not increase the sum of the costs of the edges.

4.1 Algorithm for Producing a Forest with Trees of Size at least k

The key observation is that since each partition in a k-Anonymity solution groups a vertex with at least $k-1$ other vertices, the charge of a vertex is at least equal to its distance to its $(k-1)^{st}$ nearest neighbor. The idea is to construct a directed forest such that each vertex has at most one outgoing edge and $(\overrightarrow{u,v})$ is an edge only if v is one of the $k-1$ nearest neighbors of u.

Algorithm FOREST
Invariant:

- The chosen edges do not create any cycle.

- The out-degree of each vertex is at most one.

1. Start with an empty edge set so that each vertex is in its own connected component.
2. Repeat until all components are of size at least k:

   Pick any component T having size smaller than k. Let u be a vertex in T without any outgoing edges. Since there are at most $k-2$ other vertices in T, one of the $k-1$ nearest neighbors of u, say v, must lie outside T. We add the edge $(\overrightarrow{u,v})$ to the forest. *Observe that this step does not violate any of the invariants.*

Lemma 3 *The forest produced by algorithm* FOREST *has minimum tree size at least k and has cost at most OPT.*

Proof It is evident from the algorithm description that each component of the forest it produces has at least k vertices.

Let the cost of an edge $(\overrightarrow{u,v})$ be paid by vertex u. Note that each vertex u pays for at most one edge to one of its $k-1$ nearest neighbors. As noted earlier, this is less than the charge of this vertex in any k-Anonymity solution. Thus, the sum of costs of all edges in the forest is less than OPT, the total charge of all vertices in an optimal solution. ■

In what follows we consider the underlying undirected graph on the edges.

4.2 Algorithm to Decompose Large Components into Smaller Ones

We next show how to break any component with size greater than $\max\{2k-1, 3k-5\}$ into two components each of size at least k. Let the size of the component we are breaking be $s > \max\{2k-1, 3k-5\}$.

Algorithm DECOMPOSE-COMPONENT

1. Pick any vertex u as the candidate vertex.
2. Root the tree at the candidate vertex u. Let U be the set of subtrees rooted at the children of u. Let the size of the largest subtree of u be ϕ, rooted at vertex v. If $s-\phi \geq k-1$, then we do one of the following partition and terminate (see Figure 3).

   A. If $\phi \geq k$ and $s-\phi \geq k$, then partition the tree into the largest subtree and the rest.

   B. If $s-\phi = k-1$, partition the tree into a component containing the subtrees rooted at the children of v and the rest. To connect the children of v create a dummy vertex v' to replace v. Note that v' is only a Steiner vertex (see Figure 4) and does not contribute to the size of the first component. Clearly, the sizes of both the components are at least k.

   C. If $\phi = k-1$, then partition into a component containing the subtree rooted at v along with the vertex u and the rest. In order to connect the children of u in the second component, we create a Steiner vertex u'.

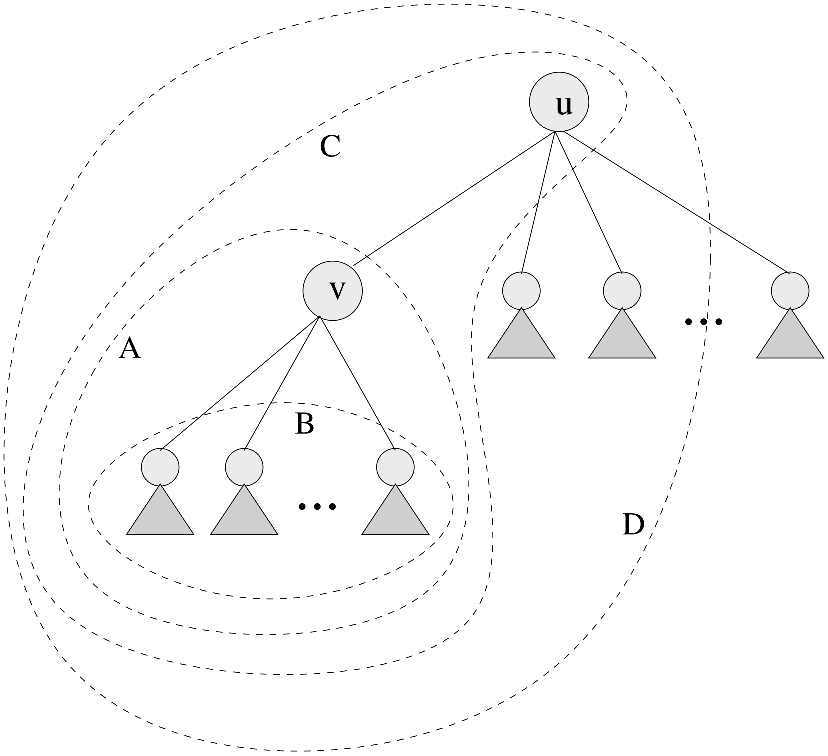


Figure 3: The decompositions corresponding to the sub-cases of the algorithm DECOMPOSE-COMPONENT.

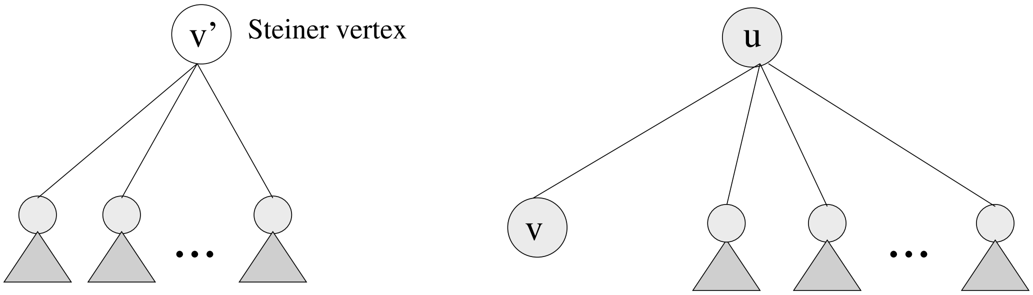


Figure 4: The decomposition corresponding to case B; the left partition contains a Steiner vertex v' that does not contribute to its size.

D. Otherwise, all subtrees have size at most $k - 2$. In this case, we create an empty partition and keep adding subtrees of u to it until the first time its size becomes at least $k - 1$. Clearly, at this point, its size is at most $2k - 4$. Put the remaining subtrees (containing at least $k - 1$ vertices, since there are at least $3k - 4$ vertices in all) into the other partition. Observe that since $s \geq 2k$, at most one of the partitions has size equal to $k-1$. If such a partition exists, add u to that partition, else add u to the first partition. In order to keep the partition not containing u connected, a Steiner vertex u' corresponding to u is placed in it.

3. Otherwise, pick the root of the largest subtree v as the new candidate vertex and go to Step 2.

Lemma 4 *The above algorithm terminates.*

Proof We will prove this by showing that the size of the largest component ϕ (in Step 2) decreases in each iteration. Consider moving from candidate vertex u in one iteration to candidate vertex v in the next iteration. Since the algorithm did not terminate with u, if we root the tree at v, then the size of the subtree rooted at u is less than $k-1$. When we consider the largest subtree under v, either it is rooted at u, in which case, it is smaller than $k-1 < s-(k-1)$ and the algorithm terminates in this step; otherwise, the new largest subtree is a subtree of the previous largest subtree. ■

Theorem 5 *There is a polynomial-time algorithm for the k-Anonymity problem, that achieves an approximation ratio of $\max\{2k-1, 3k-5\}$.*

Proof First, use Algorithm FOREST to create a forest with cost at most OPT and minimum tree size at least k. Then repeatedly apply Algorithm DECOMPOSE-COMPONENT to any component that has size larger than $\max\{2k-1, 3k-5\}$. Note that both these algorithms terminate in $O(kn^2)$ time. ■

The above algorithm can also be used when the attributes are assigned weights and the goal is to minimize the weighted generalization cost. In this case, the cost contributed by an attribute to an edge in the graph G is multiplied by its weight. The rest of the algorithm proceeds as before. It is also easy to extend the above analysis to the version of the problem where we allow an entire row to be deleted from the published database, instead of forcing it to pair with at least $k-1$ other rows. The deletion of an entire row is modeled as suppressing all the entries of that row (or generalizing all the entries of that row to the highest level). The objective function is the same as before: minimize the overall generalization cost. We first note that the distance between any two vertices is no more than the cost of deleting a vertex. Thus, if we run the same algorithm as above, the total cost of the forest F produced by Algorithm FOREST is no more than the optimal k-Anonymity cost (this is because the charge of any vertex in the optimal k-Anonymity solution is still no less than its distance to its $(k-1)^{st}$ nearest neighbor). The analysis for the rest of the algorithm remains the same.

5. Improved Algorithm for 2-Anonymity

In this section, we study the special case of $k = 2$. The algorithm of the previous section gives a 3-approximation algorithm for this case. We improve upon this result for binary alphabet, and provide a polynomial-time 1.5-approximation algorithm for 2-Anonymity (note that for binary alphabet, generalization is equivalent to suppression). This algorithm uses a technique that is completely different from the previous algorithm, and could potentially be extended to get an improved approximation factor for the general case. For this algorithm, we use the minimum-weight $[1, 2]$-factor of a graph constructed from the 2-Anonymity instance. A $[1, 2]$-factor of an edge-weighted graph G is defined to be a spanning (i.e., containing all the vertices) subgraph F of G such that each vertex in F has degree 1 or 2. The weight of F is the sum of the weights of the edges in F. Cornuejols (1988) showed that a minimum-weight $[1, 2]$-factor of a graph can be computed in polynomial time.

Given an instance of the 2-Anonymity problem on binary alphabet, we create an edge-weighted complete graph $G = (V, E)$ as follows. The vertex set V contains a vertex corresponding to each vector in the 2-Anonymity problem. The weight of an edge (a, b) is the Hamming distance between the vectors represented by a and b (i.e., the number of positions at which they differ). First we obtain a minimum-weight $[1, 2]$-factor F of G. By optimality, F is a vertex-disjoint collection of edges and pairs of adjacent edges (if a $[1, 2]$-factor has a component which is either a cycle or a path of length ≥ 3, we can obtain a $[1, 2]$-factor of smaller weight by removing edge(s)). We treat each component of F as a *cluster*, i.e., retain the bits on which all the vectors in the cluster agree and replace all other bits by $*$s. Clearly, this results in a 2-anonymized table.

Theorem 6 *The number of $*$s introduced by the above algorithm is at most 1.5 times the number of $*$s in an optimal 2-Anonymity solution.*

Before we prove this theorem, consider three m-bit vectors x_1, x_2 and x_3 with pairwise Hamming distances α, β and γ as shown in Figure 5. Without loss of generality, let $\gamma \geq \alpha, \beta$. Let x_{med} denote the *median* vector whose i^{th} bit is the majority of the i^{th} bits of x_1, x_2 and x_3 and let p, q and r be the Hamming distances to x_{med} from x_1, x_2 and x_3 respectively. Let x_s be the *star* vector obtained by minimal suppression of x_1, x_2 and x_3, i.e., it has the common bits where the three vectors agree and $*$s elsewhere. Observe that $\alpha = q + r$, $\beta = r + p$ and $\gamma = p + q$. The other relevant distances are shown in the figure.

Observation 7 *If vertices x_1, x_2 and x_3 (as shown in Figure 5) form a cluster in a k-Anonymity solution, the number of $*$s in each modified vector is exactly equal to $p + q + r = \frac{1}{2}(\alpha + \beta + \gamma)$. If the cluster contains additional vertices, then the number of $*$s is at least $\frac{1}{2}(\alpha + \beta + \gamma)$.*

To see this, first note that since x_{med} is the median vertex, the attributes that contribute to p, q and r are distinct. Therefore, the number of $*$s in each modified vector is at least $p + q + r$. Moreover, when x_1, x_2 and x_3 are the only three vertices in the cluster, each attribute corresponding to a $*$ in the modified vector contributes to exactly one of p, q and r.

Let c_{OFAC} denote the weight of an optimal $[1, 2]$-factor, let c_{ALG} be the cost of the 2-Anonymity solution obtained from it and let OPT denote the cost of the optimal 2-Anonymity solution respectively. The optimal 2-Anonymity solution can be assumed to consist only of disjoint clusters of size 2 or 3 (as bigger clusters can be broken into such clusters without increasing the cost). We can derive a $[1, 2]$-factor from this solution as follows: for each cluster of size 2, include the edge between the two vertices; for a cluster of size 3, include the two lighter edges of the triangle formed by the three vertices. Denote the weight of this $[1, 2]$-factor by c_{FAC}.

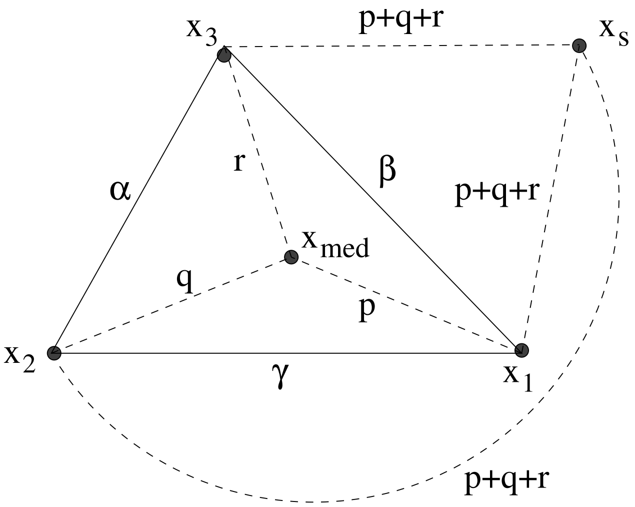


Figure 5: Three vectors and their corresponding “median” and “star” vectors

Lemma 8 $c_{ALG} \leq 3 \cdot c_{OFAC}$

Proof Consider the optimal $[1, 2]$-factor and the k-Anonymity solution corresponding to it. For a cluster of size 2, we have to suppress all the bits at which the two vectors differ so that the total number of $*$s in the two rows is twice the Hamming distance (which is equal to the edge weight). For a cluster of size 3, say the one in the figure, by Observation 7, the number of $*$s in each row is exactly $(\alpha + \beta + \gamma)/2$. So, the total number of stars is $\frac{3}{2}(\alpha + \beta + \gamma) \leq 3(\alpha + \beta)$ (using triangle inequality). The optimal $[1, 2]$-factor would have contained the two lighter edges of the triangle, incurring a cost of $(\alpha + \beta)$ for this cluster. Summing over all the clusters formed by the optimal $[1, 2]$-factor algorithm, we get $c_{ALG} \leq 3 \cdot c_{OFAC}$. ■

Lemma 9 $c_{FAC} \leq \frac{1}{2}OPT$

Proof Consider the optimal k-Anonymity solution and the $[1, 2]$-factor corresponding to it. For a cluster of size 2, cost incurred by the $[1, 2]$-factor FAC is equal to half the cost incurred in OPT. For a cluster of size 3, say the one in Figure 5, cost incurred in FAC is equal to $\alpha+\beta \leq \frac{2}{3}(\alpha+\beta+\gamma) = \frac{4}{3}(p + q + r)$, where the inequality is obtained by using the fact $\gamma \geq \alpha, \beta$. Since the cost incurred in OPT is $3(p + q + r)$, cost incurred in FAC is at most half the cost incurred in OPT. By summing over all the clusters, we get $c_{FAC} \leq OPT/2$. ■

Since $c_{OFAC} \leq c_{FAC}$, it follows from the above lemmas that $c_{ALG} \leq \frac{3}{2}OPT$, which proves Theorem 6. For an arbitrary alphabet size, x_{med} is no longer defined. However, it can be shown that $OPT \geq (\alpha+\beta+\gamma) \geq \frac{3}{2}(\alpha+\beta)$, proving $c_{FAC} \leq \frac{2}{3}OPT$. Since $c_{ALG} \leq 3 \cdot c_{OFAC}$ holds as before, we get $c_{ALG} \leq 2 \cdot OPT$. Thus, the same algorithm achieves a factor 2 approximation for 2-Anonymity with Suppression for arbitrary alphabet size.

6. Improved Algorithm for 3-Anonymity

We now present a 2-approximation algorithm for 3-Anonymity with a binary alphabet (again generalization is equivalent to suppression in this case). The idea is similar to the algorithm for 2-Anonymity. We construct the graph G corresponding to the 3-Anonymity instance as in the previous algorithm. A 2-factor of a graph is a spanning subgraph with each vertex having degree 2 (in other words, a collection of vertex-disjoint cycles spanning all the vertices). We first run the polynomial-time algorithm to find a minimum-weight 2-factor F of the graph G (Cornuejols, 1988). We show that the cost of this 2-factor, say c_{OFAC}, is at most $2/3$ times the cost of the optimal 3-Anonymity solution, say OPT. Then, we show how to transform this 2-factor F into a 3-Anonymity solution ALG of cost $c_{ALG} \leq 3 \cdot c_{OFAC}$, giving us a factor-2 approximation algorithm for 3-Anonymity.

Lemma 10 *The cost of the optimal 2-factor, c_{OFAC} on graph G corresponding to the vectors in the 3-Anonymity instance is at most $\frac{2}{3}$ times the cost of the optimal 3-Anonymity solution, OPT.*

Proof Consider the optimal 3-Anonymity solution. Observe that it will cluster 3, 4 or 5 vertices together (any larger groups can be broken up into smaller groups of size at least 3, without increasing the cost of the solution). Given an optimal solution to the 3-Anonymity problem, we construct a 2-factor solution as follows: for every cluster of the 3-Anonymity solution, pick the minimum-weight cycle involving the vertices of the cluster. Next, we analyze the cost c_{FAC} of this 2-factor. Define the *charge* of a vertex to be the number of $*$s in the vector corresponding to this vertex in the 3-Anonymity solution. We consider the following three cases:

(a) If a cluster i is of size 3, the 2-factor contains a triangle on the corresponding vertices. Let a, b and c be the lengths of the edges of the triangle. By Observation 7, we get that $(a+b+c)$ is twice the charge of each vertex in this cluster. Thus, OPT pays a total cost of $OPT_i = \frac{3}{2}(a+b+c)$ while FAC pays $c_{FAC,i} = a+b+c = \frac{2}{3}OPT_i$.

(b) If a cluster i is of size 4, the 2-factor corresponds to the cheapest 4-cycle on the four vertices. Let τ be the sum of the weights of all the $\binom{4}{2} = 6$ edges on these four vertices. Consider the three 4-cycles on these vertices. As each edge appears in two 4-cycles, the average cost of a 4-cycle is $\frac{2}{3}\tau$. By choosing the minimum weight 4-cycle, we ensure that the cost paid by FAC for these vertices $c_{FAC,i} \leq \frac{2}{3}\tau$. Also, by Observation 7, the charge of any of these 4 vertices is at least half the cost of any triangle on (three of) these four vertices. The cost of the most expensive triangle is at least equal to the average cost over all the $\binom{4}{3} = 4$ triangles, which is equal to $\frac{2}{4}\tau$ (since each edge appears in two triangles). Hence the cost paid by OPT, $OPT_i \geq 4 \cdot \frac{1}{2} \cdot \frac{2}{4} \cdot \tau = \tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

(c) If a cluster i is of size 5, let τ be the sum of weights of all $\binom{5}{2} = 10$ edges on these five vertices. By an argument similar argument to (b), FAC pays $c_{FAC,i} \leq \frac{5}{10}\tau$. Also, the charge of any of these vertices is at least half the cost of any triangle on (three of) these vertices. Since the average cost of a triangle is $\frac{3}{10}\tau$, the number of $*$s in each vertex is at least $\frac{1}{2}\frac{3}{10}\tau$. Thus, cost paid by OPT for cluster i, $OPT_i \geq 5 \cdot \frac{1}{2} \cdot \frac{3}{10} \cdot \tau = \frac{3}{4}\tau$. Thus, $c_{FAC,i} \leq \frac{2}{3}OPT_i$.

Thus, adding up over all clusters, we get $c_{FAC} \leq \frac{2}{3}OPT$. Thus, $c_{OFAC} \leq \frac{2}{3}OPT$. ■

Lemma 11 *Given a 2-factor F with cost c_F, we can get a solution for 3-Anonymity of cost $c_{ALG} \leq 3 \cdot c_F$.*

Proof To get a solution for 3-Anonymity, we make every cycle in F with size 3, 4 or 5 into a cluster. Let $\text{len}(C)$ denote the length of a cycle C in the 2-factor. For each cycle larger C, if $\text{len}(C) = 3x$ for x an integer, then we decompose it into x clusters, each containing 3 adjacent vertices of C. Similarly, if $\text{len}(C) = 3x+1$, x an integer, we decompose it into x clusters: $x-1$ of size 3, and one of size 4. If $\text{len}(C) = 3x+2$, x an integer, then we decompose it into $x-2$ clusters of size 3, and two clusters of size 4. In all these cases, of all the possible decompositions, we choose the one in which the total cost of edges of the cycle within the clusters is minimized. Depending on the size of the cycle C in the 2-factor, we can show that the 3-Anonymity solution ALG pays as follows:

(a) For a triangle, ALG pays $3 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(b) For a 4-cycle, ALG pays at most $4 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

(c) For a 5-cycle, ALG pays at most $5 \cdot \frac{1}{2}\text{len}(C) \leq 3 \cdot \text{len}(C)$.

The above inequalities follow from an observation similar to Observation 7, namely that the vertices of a cycle C can differ in at most $\frac{1}{2}\text{len}(C)$ attributes.

(e) For a $(3x+1)$-cycle, $x > 1$, ALG pays at most $\frac{6(x-1)+12}{3x+1} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by considering the minimum 3-Anonymity cost over the $(3x+1)$ possible decompositions into clusters. Each edge e of the cycle C appears in a cluster of size 4 in three decompositions and contributes a cost of at most $4w(e)$ to the k-Anonymity cost of the decomposition. In addition, each edge appears in a cluster of size 3 in $(2(x-1))$ decompositions contributing a cost of at most $3w(e)$ to the k-Anonymity cost of these decompositions. Summing over all edges, the total k-Anonymity cost of all the $3x+1$ decompositions is at most $(3 \cdot 2(x-1) + 4 \cdot 3) \cdot \text{len}(C)$ and ALG pays no more than the average cost of a decomposition.

(f) For a $(3x+2)$-cycle, $x > 1$, ALG pays at most $\frac{6(x-2)+24}{3x+2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. This is obtained by an analysis similar to (e) above. Note that we get a better bound on the cost by splitting into $x-2$ clusters of size 3 and two clusters of size 4, instead of $x-1$ clusters of size 3 and one clusters of size 5.

Thus, summing over all clusters, ALG pays no more than three times the total cost of all cycles, i.e., $c_{ALG} \leq 3 \cdot c_F$. ■

Note that the above analysis is tight, since equality can hold in case (f), when $x = 2$, e.g. for vectors $\{0000, 0001, 0011, 0111, 1111, 1110, 1100, 1000\}$, where the optimal 2-factor is a cycle through all the vertices in the given order.

Combining the above lemmas, we obtain a factor 2 approximation for 3-Anonymity.

7. Conclusion and further research directions

We show that the k-Anonymity problem is NP-hard even when the attribute values are ternary and we are allowed only to suppress entries. Then we gave an $O(k)$-approximation algorithm for k-Anonymity with Generalization for arbitrary k and arbitrary alphabet size. For a binary alphabet, we provided improved approximation algorithms for $k = 2$ (an approximation factor of 1.5) and $k = 3$ (an approximation factor of 2). We also showed that for k-Anonymity, it is not possible to achieve an approximation factor better than $k/4$ by using the graph representation. It would also be interesting to see a hardness of approximation result for k-Anonymity without assuming the graph representation.

Releasing a database after k-anonymization prevents definitive *record linkages* with publicly available databases (Sweeney, 2002). In particular, for each record in the public database, at least k records in the k-anonymized database could correspond to it, which hides each individual in a crowd of k other people. The privacy parameter k must be chosen according to the application in order to ensure the required level of privacy. One source of concern about the k-anonymization model is that for a given record in the public database, all the k records corresponding to it in the anonymized database might have the same value of the sensitive attribute(s) ("Diseases" in our examples), thus revealing the sensitive attribute(s) conclusively. To address this issue, we could add a constraint that specifies that for each cluster in the k-anonymized database, the sensitive attribute(s) should take at least r distinct values. Recently Machanavajjhala, Kifer, Gehrke, and Venkitasubramaniam (2006) propose imposing additional constraints that there be a good representation of sensitive attributes for each block of k-anonymized records.

Another interesting direction of research is to extend the basic k-Anonymity model to deal with changes in the database. A hospital may want to periodically release an anonymized version of its patient database. However, releasing several anonymized versions of a database might leak enough information to enable *record linkages* for some of the records. It would be useful to extend the k-Anonymity framework to handle inserts, deletes and updates to a database.

References

G. Aggarwal, T. Fèder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, and A. Zhu. Anonymizing tables. In Thomas Eiter and Leonid Libkin, editors, *Proceedings of the 10th International Conference on Database Theory*, volume 3363 of *Lecture Notes in Computer Science*, pages 246–258. Springer, January 2005.

G. Aggarwal, N. Mishra, and B. Pinkas. Secure computation of the k^{th}-ranked element. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 44–55, 2004.

D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *Proceedings of the 20th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 247–255, 2001.

R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 439–450, May 2000.

R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 251–262, 2005.

A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, June 2005.

S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Proceedings of the 2nd Theory of Cryptography Conference*, pages 363–385, 2005.

G.P. Cornuejols. General factors of graphs. *Journal of Combinatorial Theory*, 45:185–198, 1988.

I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 202–210, 2003.

C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *Advances in Cryptology: Proceedings of Crypto*, pages 528–544, 2004.

A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proceedings of the 22nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 211–222, June 2003.

M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology: Proceedings of Eurocrypt*, pages 1–19, 2004.

O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

V. Kann. Maximum bounded H-matching is MAX SNP-complete. *Information Processing Letters*, 49:309–318, 1994.

K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Proceedings of the 24th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 118–127, June 2005.

J. Kleinberg, C. Papadimitriou, and P. Raghavan. Auditing boolean attributes. *Journal of Computer and System Sciences*, 6:244–253, 2003.

Y. Lindell and B. Pinkas. Privacy preserving data mining. *Journal of Cryptology*, 15(3):177–206, 2002.

A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-Diversity: Privacy beyond k-Anonymity. In *Proceedings of the 22nd International Conference on Data Engineering*, 2006.

A. Meyerson and R. Williams. On the complexity of optimal k-Anonymity. In *Proceedings of the 23rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, pages 223–228, June 2004.

P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, 13(6):1010–1027, 2001.

P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proceedings of the 17th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, page 188, 1998.

L. Sweeney. Uniqueness of simple demographics in the U.S. population. Technical Report LIDAP-WP4, Laboratory for International Data Privacy, Carnegie Mellon University, Pittsburgh, PA, 2000.

L. Sweeney. k-Anonymity: A model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5):557–570, 2002.

A. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167, 1986.