Anonymizing Tables

G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani,
R. Panigrahy, D. Thomas, A. Zhu

Stanford University*

Abstract. We consider the problem of releasing tables from relational database 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 for each person contained in the release cannot be distinguished from at least $k-1$ other persons whose information also appears in the release. In the k-ANONYMITY *problem* the objective is to minimally suppress cells in the table so as to ensure that the released version is k-anonymous. We show that the k-ANONYMITY problem is NP-hard even when the attribute values are ternary. On the positive side, we provide an $O(k)$-approximation algorithm for the problem. This improves upon the previous best-known $O(k \log k)$-approximation. We also give improved positive results for the interesting special cases with small 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.

1 Introduction

The information age has witnessed a tremendous growth in the amount of personal data that can be collected and analyzed. This has led to an increasing use of data mining tools with the basic goal of inferring trends in order to predict the future. However, concerns regarding conflicts of this goal with the desire for privacy of personal data have emerged [Eur98,Tim97]. In many scenarios, access to large amounts of *personal data* is essential in order for accurate inferences to be drawn. For example, hospitals might wish to collaborate in order to catch the outbreak of epidemics in their early stages. This requires them to allow outside access to medical records of their patients. In such cases, one would like to provide data in a manner that enables one to draw inferences without violating the privacy of individual records.

Different approaches to address this problem have emerged recently. One approach is to use *perturbation* techniques in order to hide the exact

* Supported in part by NSF Grants IIS-0118173, EIA-0137761, and ITR-0331640, and grants from Microsoft, SNRC, and Veritas.

values of the data [AS00,AA01,DN03,EGS03,AKSX03,KPR00]. However, this may not be suitable if one wants to make inferences with 100% confidence. If the function to be evaluated is known in advance, we can use techniques from *secure multi-party computation* [LP00,AMP04,FNP04]. However interactive data-mining tasks are inherently ad-hoc and the queries are not known ahead of time.

Another approach is to *suppress* or *generalize* some of the sensitive data values. We consider the k-anonymity model which was proposed by Samarati and Sweeney [Swe02,SS98]. Suppose we have a table with n tuples and m attributes. Let $k > 1$ be an integer. We wish to release a modified version of this table, where we can suppress the values of certain cells in the table. *The objective is to minimize the number of cells suppressed while ensuring that for each tuple in the modified table, there are $k-1$ other tuples in the modified table identical to it.* 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	94305	Common Cold
35	White	Female	94045	Flu
27	Black	Female	92010	Flu
27	White	Female	92010	Hypertension

By combining with information available in public databases (such as 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 [Swe00] 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. By combining publicly released de-identified health insurance data with the voter database, Sweeney [Swe02] could identify the medical records of the governor of Massachusetts. Hence releasing the above de-identified table could lead to violation of privacy. Instead we would like to suppress some of these entries so that any (Age, Race, Gender, Zip Code) tuple corresponds to at least k individuals. For instance, when $k = 2$, we could obtain the following anonymized table.

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

We study the problem of finding the *optimal* (in terms of minimizing the number of cells suppressed) k-anonymized table for any given table instance. We show that this problem is NP-hard even for the special case of ternary attribute values. This significantly strengthens the NP-hardness result in [MW04], which required the domain of attribute values to be larger than the number of tuples in the table. On the positive side, we give an $O(k)$-approximation algorithm for the k-Anonymity problem (for arbitrary alphabet size) using a graph representation. This improves upon the previous best-known approximation guarantee of $O(k \log k)$ [MW04]. We note that it is not possible to achieve an approximation factor better than $O(k)$ with the graph representation of the problem (we elaborate more on this later). In addition, for binary alphabets, we give a 1.5-approximation algorithm for $k = 2$ and a 2-approximation algorithm for $k = 3$.

The rest of the paper is organized as follows. In Section 2, we specify our model and formally state the problem of k-Anonymity. We establish the NP-hardness of k-Anonymity in Section 3. Then, we provide a 1.5 approximation algorithm for the 2-Anonymity problem for binary alphabet in Section 4, and follow this up with a brief sketch of the 2-approximation algorithm for 3-Anonymity (for binary alphabet) in Section 5. In Section 6, we show a lower bound of $O(k)$ on the approximation ratio for the graph representation of the k-Anonymity problem and then present a tight $O(k)$-approximation algorithm for the same. In the Appendix, we present the details of the 2-approximation algorithm for 3-Anonymity.

2 Model and Main results

Consider a database with n rows and m columns in which each entry comes from a finite alphabet Σ. For example, in a medical database, the rows represent individuals and the columns correspond to the different attributes. We would like to suppress some of the entries so that each row becomes identical to at least $k-1$ other rows. A suppressed entry is denoted by the symbol $*$. Since suppression results in the release of *less information and hence less utility*, we would like to suppress as few entries as possible.

We can view the database as consisting of n m-dimensional vectors: $x_1, x_2, \ldots, x_n \in \Sigma^m$. A *$k$-anonymous suppression* function t maps each x_i to $\tilde{x}_i$ by replacing some components of x_i by $*$, so that every $\tilde{x}_i$ is identical to at least $k-1$ other $\tilde{x}_j$'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 $*$'s in all the $\tilde{x}_i$'s.

k-ANONYMITY: *Given $x_1, x_2, \ldots, x_n \in \Sigma^m$, obtain a suppression function t so that $c(t)$ is minimized.*

We show that k-ANONYMITY is NP-hard even when the alphabet size $|\Sigma| = 3$. This improves upon the NP-hardness result of [MW04] which required an alphabet size of n. On the positive side, we provide an $O(k)$-approximation algorithm for arbitrary k and arbitrary alphabet size. For a binary alphabet, we also provide 1.5-approximation for $k = 2$ and 2-approximation for $k = 3$.

3 NP-hardness of *k*-ANONYMITY

Theorem 1. k-ANONYMITY *is NP-hard for a ternary alphabet* ($\Sigma = \{0, 1, 2\}$).

Proof Sketch: We show the NP-hardness for $k = 3$ by reduction from EDGE PARTITION INTO TRIANGLES [Kan94]: *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, we create a database as follows. W.l.o.g., we assume that G is simple. The rows correspond to the $3m$ edges and the columns to the n vertices of G. The row corresponding to edge (a, b), r_{ab}, has 1's in the positions corresponding to a and b and 0's everywhere else. We first show that the cost of the optimal 3-ANONYMITY solution is at most $9m$ if and only if E can be partitioned into a collection of m disjoint triangles and 4-stars[1].

Suppose such a partition of edges is given. Consider any triangle (with a, b, c as the vertices). By suppressing the positions a, b and c in the rows r_{ab}, r_{bc} and r_{ca}, we get a cluster with three $*$'s in each modified row. Similarly, 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 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 $9m$. Thus the

[1] By 4-star, we mean a tree on four vertices with a vertex of degree 3.

solution cost is exactly $9m$ and each modified row has exactly three $*$'s. Since any cluster of size > 3 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 a triangle or a 4-star. 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. This corresponds to a partition of the graph edges into triangles or 4-stars, instead of only triangles.

Since we want a reduction from EDGE PARTITION INTO TRIANGLES, we "force" the 4-stars to pay more $*$'s by increasing the number of columns created in our k-ANONYMITY instance. Let $t = 1 + \lceil \log_2 3m \rceil$. Consider an arbitrary ordering of E and express the rank of an edge $e = (a, b)$ in binary notation as $b_1 b_2 \ldots b_t$. Every row in the database now has t *blocks*, each of which has n columns. In the row corresponding to edge e, each block has zeros in all positions except a and b. Depending on the values in positions a and b, a block can be in two configurations: $conf_0$ has 1 in position a and 2 in position b while $conf_1$ has 2 in position a and 1 in position b. The i^{th} block in this row has configuration $conf_{b_i}$. (For example, consider a complete graph on four vertices: $\{v_1, v_2, v_3, v_4\}$. Suppose edge (v_1, v_2) has rank $5 = (0101)_2$. The corresponding row would have 4 blocks of 4 columns each: $1200 - 2100 - 1200 - 2100$.)

We will now show that the cost of the optimal 3-ANONYMITY solution is at most $9mt$ if and only if 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. Each cluster has exactly three rows. The corresponding edges should form a triangle: As any two of these edges differ in the configuration of at least one block, there would have been more than $3t$ $*$'s per row if they formed a 4-star instead. Thus we get a partition of E into disjoint triangles.

By reduction from EDGE PARTITION INTO r-CLIQUES [Kan94], 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 $r \geq 3$. $\square$

4 Algorithm for 2-Anonymity

For a binary alphabet, we provide a polynomial time 1.5-approximation algorithm for 2-Anonymity, using a polynomial time algorithm for obtaining a minimum weight $[1, 2]$-factor of a graph. 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 edges in F. Cornuejols [Cor88] 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, 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 node-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 lesser weight $[1, 2]$-factor 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 database.

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

Proof Sketch: Let ALG and OPT denote the costs of the above solution and optimal 2-Anonymity solution respectively. Let $OFAC$ denote the weight of an optimal $[1, 2]$-factor. 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 derive a $[1, 2]$-factor from this solution as follows. Include the edge between the two vertices in each 2-cluster. For 3-clusters, include the two edges of lesser weights amongst the three edges. Denote the weight of this $[1, 2]$-factor by FAC.

Consider three m-bit vectors x_1, x_2 and x_3 with pairwise Hamming distances α, β and γ as in Fig. 1. W.l.o.g, let $\gamma \geq \alpha, \beta$. Let x_m 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_m from the three vectors. 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.

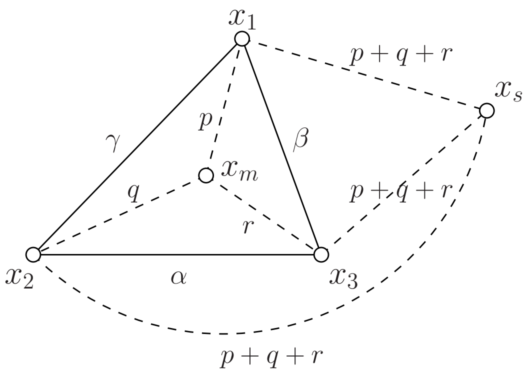

Fig. 1. Three vectors and their corresponding "median" and "star" vectors

Lemma 1. $ALG \leq 3 \cdot OFAC$

Proof Sketch: For a 2-cluster, we have to suppress all the bits at which the two vectors differ so that the total number of $*$'s is twice the Hamming distance (which is the edge weight). For a 3-cluster, say the one in the figure, the number of $*$'s is $(p + q + r)$ for each vector, so that the total is $3(p + q + r) = \frac{3}{2}(\alpha + \beta + \gamma) \leq 3(\alpha + \beta)$ (using triangle inequality). The optimal $[1, 2]$-factor would have contained two (lesser weight) edges, incurring a cost of $(\alpha + \beta)$ for this cluster. Considering all the clusters formed by the optimal $[1, 2]$-factor algorithm, we get $ALG \leq 3OFAC$. $\square$

Lemma 2. $FAC \leq OPT/2$

Proof Sketch: For a 2-cluster, cost incurred in $FAC = \frac{1}{2}$(cost incurred in OPT). For a 3-cluster (in the figure), cost incurred in $FAC = \alpha + \beta \leq \frac{2}{3}(\alpha + \beta + \gamma) = \frac{4}{3}(p + q + r)$, where the inequality follows using $\gamma \geq \alpha, \beta$. Since the cost incurred in OPT is $3(p+q+r)$, cost incurred in $FAC < \frac{1}{2}$(cost incurred in OPT). By considering all the clusters, we get $FAC \leq OPT/2$. $\square$

Since $OFAC \leq FAC$, it follows from the above lemmas that $ALG \leq \frac{3}{2}OPT$. $\square$

For an arbitrary alphabet size, x_{maj} is no longer defined. However as before it can be shown that $OPT \geq (\alpha+\beta+\gamma) \geq \frac{3}{2}(\alpha+\beta)$, proving $FAC \leq$

$\frac{2}{3}OPT$. As $ALG \leq 3OFAC$ holds as before, we get $ALG \leq 2OPT$. Thus the algorithm achieves a factor 2 approximation for arbitrary alphabet size.

5 Algorithm for 3-Anonymity

We now present a 2-approximation algorithm for 3-Anonymity with a binary alphabet. 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 node-disjoint cycles spanning all the vertices). We run the polynomial time algorithm to find a minimum-weight 2-factor F of the graph G [Cor88]. We first show that the cost of this 2-factor, say $OFAC$, 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 of cost at most $3OFAC$, giving us a factor-2 approximation algorithm for 3-Anonymity. The details can be found in the appendix.

6 Algorithm for general k-Anonymity

In this section, we address the problem of k-Anonymity for general k and arbitrary alphabet size, and give an $O(k)$-approximation algorithm for the problem. Given an instance of the k-Anonymity problem, we create an edge-weighted graph $G = (V, E)$. The vertex set V contains a vertex corresponding to each vector in the k-Anonymity problem. The weight, $w(e)$ of an edge $e = (a, b)$ is the number of attributes along which the vectors represented by a and b differ.

As mentioned in the introduction, with this representation, we lose some information about the structure of the problem, and cannot achieve a better than $O(k)$ approximation factor for the k-anonymity problem. We show this by giving two instances whose k-anonymity cost differs by a factor of $O(k)$, 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 kl. 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 $4l$. Note that the graph corresponding to both the instances is a k-clique with the all pairwise distances being $2l = 2^{k-1}$. Hence it is not possible to improve on the $O(k)$ approximation for the edge-weighted graph representation of the database instance. Note that the 1.5 approximation for 2-anonymity and 2-approximation for 3-anonymity, in the case of binary alphabet, used more than the edge-weighted graph representation of the database instance.

Next, we describe our $O(k)$-approximation algorithm for the k-anonymity problem.

For any given k-ANONYMITY solution, define the *charge* of a vertex to be the number of $*$'s introduced into the vector it represents. Let OPT denote the cost of an optimal k-ANONYMITY solution, i.e., OPT is the sum of charges of all vertices in an optimal k-ANONYMITY solution.

Let $F = \{T_1, T_2, \ldots, T_r\}$, a forest 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 at most the cost of the tree, $W(T_i) = \Sigma_{e \in E(T_i)} w(e)$. This is because any attribute along which a pair of vertices differs appears on the path between the two vertices. Thus, the k-anonymity cost of such a partition 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 cost of a forest is simply the sum of the costs of its trees. The ratio of the k-anonymity cost to the simple cost of a forest is at most the number of vertices in the largest tree in the forest. Thus, if we can find a forest with the size of the largest component at most L and cost at most OPT, then we have an L approximation algorithm. Next, we present an algorithm that finds such a forest with $L \leq 3k - 3$. Actually, the forest that we obtain has dummy vertices that act as Steiner points, but this does not affect the result.

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

Outline:

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 (we are allowed to delete edges) such that each component has between k and $3k - 3$ vertices. The decomposition is done in a way that does not increase the sum of the costs of the edges.

6.1 Algorithm for producing a forest with components 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 Sketch: 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 the total charge of all vertices in an optimal solution. □

6.2 Algorithm to decompose large components into smaller ones

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

Algorithm Decompose-Component

1. Pick any vertex 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. If $\phi \leq s - k$, then we do the following partition and terminate.

   If $\phi \geq k$, then partition the tree into the largest subtree and the rest. Clearly, the size of both components is at least k. Otherwise, all subtrees have size at most $k - 1$. In this case, keep adding subtrees to a partition till the first time its size becomes at least k. Clearly, at this point, its size is at most $2k - 2$. Put the remaining subtrees and u (at least k vertices in all) into the other partition. In order to keep the first partition connected, a dummy vertex corresponding to u is placed in the first partition which acts only as a Steiner point and does not contribute to the size of the component.
3. Otherwise, pick the root of the largest subtree as the new candidate vertex and go to Step 2.

Lemma 4. *The above algorithm terminates.*

Proof Sketch: We will show 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 $s - (s - k) = k$. When we consider the largest subtree under v, either it is rooted at u, in which case, it is smaller than $k < s - k$. Otherwise, the new largest subtree is a subtree of the previous largest subtree. □

Theorem 3. *There is a* $(3k - 3)$*-approximation algorithm for the* k-ANONYMITY *problem.*

Proof Sketch: 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 $3k - 3$. □

This factor can be improved to $\max(2k - 1, 3k - 5)$ by appropriately choosing the partition for u in Step 2 of Algorithm DECOMPOSE-COMPONENT. This reduces to $3k - 5$ for $k \geq 4$.

We can easily extend the above algorithm and 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 cost of deleting an entire row is modeled as changing all the entries of that row to stars, while the objective remains to minimize the number of stars.

7 Conclusion and Open Problems

We showed that the k-Anonymity problem is NP-hard, even when the attribute values are ternary. Then we gave an $O(k)$-approximation algorithm for the general k-Anonymity problem with arbitrary alphabet size, improving upon the previous best known $O(k \log k)$-approximation. For binary alphabets, we achieve an approximation factor of 1.5 for $k = 2$ and a factor of 2 for $k = 3$.

An interesting open problem is to try to extend our $O(k)$ hardness of approximation for k-Anonymity without assuming the graph representation. Another interesting direction of research is to extend the basic k-Anonymity model to account for changes in the database. A hospital may want to periodically release an anonymized version of its patient database. How do we extend the k-Anonymity framework to handle inserts, deletes and updates to the database?

References

[AA01] D. Agrawal and C. Aggarwal. On the design and quantification of privacy preserving datamining algorithms. In *Proc. of the ACM Symp. on Principles of Database Systems*, 2001.

[AKSX03] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. Hippocratic databases. In *Proc. of the Intl. Conf. on Very Large Data Bases*, August 2003.

[AMP04] G. Aggarwal, N. Mishra, and B. Pinkas. Privacy preserving computation of the k-th ranked element. In *EUROCRYPT*, 2004.

[AS00] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 439–450, May 2000.

[Cor88] G. P. Cornuejols. General factors of graphs. In *Journal of Combinatorial Theory B 45*, pages 185–198, 1988.

[DN03] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *Proc. of the ACM Symp. on Principles of Database Systems*, pages 202–210, 2003.

[EGS03] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *Proc. of the ACM Symp. on Principles of Database Systems*, June 2003.

[Eur98] European Union. *Directive on Privacy Protection*, October 1998.

[FNP04] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[Kan94] V. Kann. Maximum bounded H-matching is MAX SNP-complete. In *Information Processing Letters, 49*, pages 309–318, 1994.

[KPR00] Jon M. Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. Auditing boolean attributes. In *Symposium on Principles of Database Systems*, pages 86–91, 2000.

[LP00] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, pages 36–54, 2000.

[MW04] A. Meyerson and R. Williams. On the complexity of optimal k-anonymity. In *Proc. of the ACM Symp. on Principles of Database Systems*, June 2004.

[SS98] P. Samarati and L. Sweeney. Generalizing data to provide anonymity when disclosing information (abstract). In *Proc. of the ACM Symp. on Principles of Database Systems*, page 188, 1998.

[Swe00] L. Sweeney. Uniqueness of simple demographics in the U.S. population. In *LIDAP-WP4. Carnegie Mellon University, Laboratory for International Data Privacy, Pittsburgh, PA*, 2000.

[Swe02] L. Sweeney. k-Anonymity: A model for protecting privacy. In *International Journal on Uncertainty Fuzziness Knowledge-based Systems*, June 2002.

[Tim97] Time. *The Death of Privacy*, August 1997.

A Detailed algorithm for 3-Anonymity

Lemma 5. *The cost of the optimal 2-factor, $OFAC$ on graph G corresponding to the vectors in the 3-*Anonymity *instance is at most* $2/3$ *times the cost of the optimal 3-*Anonymity *solution, OPT.*

Proof Sketch: Note that the optimal 3-Anonymity solution 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 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. Using an argument similar to Lemma 1, we get that $(a+b+c)$ is twice the charge of each vertex in this cluster. Thus, OPT pays a total cost of $OPT_i = 3(a+b+c)/2$ while FAC pays $FAC_i = a+b+c = \frac{2}{3}OPT_i$

(b) If a cluster i is of size 4, the 2-factor corresponds to the cheapest 4-cycle on the 4 vertices. Let τ be the sum of the weights of all the $\binom{4}{2} = 6$ edges on these four vertices. Then, by considering all 4-cycles and choosing the minimum weight 4-cycle, we ensure that the cost paid by FAC for these vertices $FAC_i \leq \frac{4}{6}\tau$. Also, the charge of any of these

4 vertices is at least half the cost of any triangle on (three of) these four vertices (again by using the argument of Lemma 1). Averaging over all triangles, we get that cost paid by OPT, $OPT_i \geq 4.\frac{1}{2}.\frac{3}{6}.\tau = \tau$. Thus, $FAC_i \leq \frac{2}{3}OPT_i$.

(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. Then, FAC pays $FAC_i \leq \frac{5}{10}\tau$. Also, the charge of any of these vertices is at least half the cost of any triangle on (three of) these vertices. Averaging over all triangles, we get that cost paid by OPT for cluster i, $OPT_i \geq 5.\frac{1}{2}.\frac{3}{10}.\tau = \frac{3}{4}\tau$. Thus, $FAC_i \leq \frac{2}{3}OPT_i$.

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

Lemma 6. *Given a 2-factor F with cost FAC, we can get a solution for 3-*Anonymity *of cost $SOL \leq 3 \cdot FAC$.*

Proof Sketch: To get a solution for 3-Anonymity, we make every cycle in F with size 3, 4 or 5 into a cluster. For each larger cycle C, if $|C| = 3x$ for x an integer, then we break it up into x clusters, each containing 3 adjacent vertices of C, such that the total cost of edges of the cycle within the clusters is minimized. Similarly, if $C = 3x + 1$, x an integer, we break it into x clusters, $x - 1$ of size 3, and one of size 4. If $C = 3x + 2$, x an integer, then we break it up into $x - 2$ clusters of size 3, and two clusters of size 4.

Let $\text{len}(C)$ denote the length of a cycle C in the 2-factor. Then depending on the size of the cycle C, we can show that the 3-Anonymity solution SOL pays as follows:

(a) For a triangle, SOL pays $3 \cdot \frac{1}{2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$.
(b) For a 4-cycle, SOL pays at most $4 \cdot \frac{1}{2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$.
(c) For a 5-cycle, SOL pays at most $5 \cdot \frac{1}{2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$.

This is so (for the above cases) because the number of attributes along which the vertices differ is at most $\text{len}(C)/2$.

(d) For a $3x$-cycle, $x > 1$, SOL pays at most $3 \cdot \frac{2x}{3x} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$.
(e) For a $(3x + 1)$-cycle, $x > 1$, SOL pays at most $\frac{2(x-1)\cdot 3+3\cdot 4}{3x+1} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$.
(f) For a $(3x + 2)$-cycle, $x > 1$, SOL pays at most $\frac{2(x-2)\cdot 3+6\cdot 4}{3x+2} \cdot \text{len}(C) \leq 3 \cdot \text{len}(C)$. (Equality can hold in this case, when $x = 2$.)

Thus, adding over all clusters, SOL pays no more than 3 times the total cost of all cycles, i.e., $3 \cdot FAC$. □

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