K-Anonymization Incremental Maintenance and Optimization Techniques

Traian Marius Truta
Department of Computer Science
Northern Kentucky University
Highland Heights, KY 41099, USA
001-859-572-7551
trutat1@nku.edu

Alina Campan
Department of Computer Science
Babes-Bolyai University
Cluj-Napoca, RO-400084, Romania
0040-264-405327
alina@cs.ubbcluj.ro

ABSTRACT
New privacy regulations together with ever increasing data availability and computational power have created a huge interest in data privacy research. One major research direction is built around *k*-anonymity property, which is required for the released data. Although many *k*-anonymization algorithms exist for static data, a complete framework to cope with data evolution (a real world scenario) has not been proposed before. In this paper, we introduce algorithms for the maintenance of *k*-anonymized versions of large evolving datasets. These algorithms incrementally manage insert/delete/update dataset modifications. Our results showed that incremental maintenance is very efficient compared with existing techniques and preserves data quality. The second main contribution of this paper is an optimization algorithm that is able to improve the quality of the solutions attained by either the non-incremental or incremental algorithms.

Categories and Subject Descriptors
K.4.1 [**Computers and Society**]**:** Public Policy Issues – *privacy*.
I.5.3 [**Pattern Recognition**]: Clustering – *algorithms*.

General Terms
Algorithms, Performance, Security.

Keywords
Privacy, *k*-anonymity, incremental, optimization, clustering.

1. INTRODUCTION
To protect the privacy of individuals in the information era is challenge that our society has just begun facing. More and more *microdata* (datasets where each tuple belongs to an individual entity) are collected by different agencies. Some of these microdata need to be released, for various purposes, to other parties in a modified form (without the direct identifying information such as *SSN*, *Name*, etc.). But even altered this way, these datasets could still present vulnerabilities that can be exploited by intruders, i.e. persons whose goals are to identify specific individuals and to use the confidential information they discover for malicious purposes. The high volume and availability of released datasets together with ever increasing computational power made the protection against those vulnerabilities an increasingly difficult task.

There are several regulations related to the use and the disclosure of confidential information [6, 8]. All these regulations, together with the necessity of collecting personal information, have funneled a huge interest in privacy research. Techniques to avoid the disclosure of confidential information exist in the literature [22]. Among them, the *k*-anonymity property required for the released microdata was recently introduced [17, 18] and extensively studied [2, 11, 21, etc.]. This property requires that in the released microdata every tuple will be indistinguishable from at least (*k*-1) other tuples with respect to a subset of attributes called quasi-identifier attributes.

Various *k*-anonymization algorithms exist, which generally proceed by using generalization and suppression [17, 19]. Besides ensuring *k*-anonymity property, these algorithms must also consider minimizing one or more cost metrics between initial and released microdata. Of particular interest are the cost metrics that quantify the *information loss* [3, 5, 14, 20]. Although producing the optimal solution for the *k*-anonymity problem w.r.t. many proposed cost measures has been proved to be NP-hard [15], there are several polynomial algorithms that produce good solutions for the *k*-anonymity problem for real life datasets [1, 3, 10, 11, etc.].

Contributions All mentioned *k*-anonymization approaches assume that the processed microdata sets are static. Nevertheless, large microdata sets containing private information are time-evolving, meaning that new data are collected and added, and old data are purged. When new *k*-anonymized versions of such a dataset are prepared for release, the current solution is to re-process the entire dataset, from scratch, without relying on previous releases of the dataset. However, processing a large dataset to achieve *k*-anonymity is time-consuming.

We propose in this paper an incremental updating technique for the maintenance of *k*-anonymized versions of large evolving datasets. Essentially, the proposed technique produces a *k*-anonymized version of a dataset $I\mathcal{M} \cup \Delta^{+}\mathcal{M} - \Delta^{-}\mathcal{M}$, starting from a previous *k*-anonymized release for the dataset $I\mathcal{M}$, which is

SAC'07, March 11-15, 2007, Seoul, Korea.

updated to include the new data in the increment dataset $\Delta^+\mathcal{M}$ and to remove the obsolete dataset $\Delta^-\mathcal{M}$. The anonymization process tries to minimize information loss. As our experimental results show, for small size datasets $\Delta^+\mathcal{M}$ / $\Delta^-\mathcal{M}$, w.r.t. the initial dataset $\mathcal{IM}$ size, the incremental updating process is far more efficient than to re-process the final dataset $\mathcal{IM} \cup \Delta^+\mathcal{M} - \Delta^-\mathcal{M}$. Also, the quality of the result remains at about the same level for our approach as for a non-incremental one.

Several approaches have been proposed to model and solve *k*-anonymization as a clustering problem [1, 3]. Our incremental updating technique continues on this direction, i.e. it uses ideas from the clustering area. We evaluate the performance of our method by comparing the results it produces against the results provided by the static (non-incremental) *k*-anonymization method presented in [3]. While a similar updating method in case of only insertions has very recently been introduced [4], a complete maintenance framework of *k*-anonymized microdata that manages insert/delete/update modifications has not, to our knowledge, been reported anywhere.

An equally important contribution we present in this paper is an optimization algorithm that is able to improve the solutions attained by either the non-incremental algorithm [3] or the incremental algorithm we propose. The optimization process is highly efficient, as our experiments have shown.

The paper is structured as follows. Section 2 introduces the concepts and notations we use along the paper. Section 3 presents the incremental *I3M+* and *I3M-* algorithms for maintaining masked microdata when inserts and deletes are performed in the initial microdata. Section 4 describes the optimization algorithm proposed for improving the quality of the masked microdata. Section 5 reports experimental results for incremental and optimization algorithms. The paper ends with conclusions and future work directions in Section 6.

2. CONCEPTS AND NOTATIONS

Let $\mathcal{IM}$ be the initial microdata and $\mathcal{MM}$ be the released (a.k.a. masked) microdata. $\mathcal{IM}$ consists in a set of tuples over an attribute set. These attributes are classified into the following three categories:

- I_1, I_2,..., I_m are *identifier* attributes such as *Name* and *SSN* that can be used to identify a record.
- K_1, K_2,…, K_n are *key* or *quasi-identifier* attributes such as *ZipCode* and *Age* that may be known by an intruder.
- S_1, S_2,…, S_r are *confidential* or *sensitive* attributes such as *PrincipalDiagnosis* and *ICD9Code* that are assumed to be unknown to an intruder.

While the identifiers attributes are removed from the published microdata, the quasi-identifier and confidential attributes are usually released to the researchers. A general assumption, as noted, is that the values for the confidential attributes are not available from any external source. This assumption guarantees that an intruder can not use the confidential attributes values to increase his/her chances of disclosure. Unfortunately, an intruder may use record linkage techniques between quasi-identifier attributes and external available information to glean the identity of individuals from the masked microdata. To avoid this possibility of disclosure, one frequently used solution is to modify the initial microdata, more specifically the quasi-identifier attributes values, in order to enforce the *k*-anonymity property.

Definition 1 (*k-anonymity property*): The ***k-anonymity property*** for a masked microdata ($\mathcal{MM}$) is satisfied if every combination of quasi-identifier attribute values in $\mathcal{MM}$ occurs *k* or more times.

Generalization of the quasi-identifier attributes is one of the techniques widely used for *k*-anonymization. It consists in replacing the actual value of an attribute with a less specific, more general value that is faithful to the original [19].

Initially, this technique was used for *categorical* attributes and employed predefined (static) domain and value generalization hierarchies [19, 17, 11, 9, 12]. Generalization was extended for *numerical* attributes either by using *predefined hierarchies* [9] or a *hierarchy-free model* [11].

For categorical attributes we used generalization based on predefined hierarchies at the cell level [12]. We denote by $\mathcal{H}_C$ the hierarchies (domain and value) associated to the categorical quasi-identifier attribute *C*. For numerical attributes we use the hierarchy-free generalization [11], which consists in replacing the set of values to be generalized to the smallest interval that includes all the initial values. For instance, the values: 35, 38, 36 are generalized to the interval [35-38]. This approach is best suited to solve the *k*-anonymity problem using clustering methods, because the set of clusters are initially created, and the generalization occurs afterwards at the cluster level, and as needed for each individual cluster.

The central idea in employing clustering to solve the *k*-anonymization problem is as follows. A clustering method is applied on microdata, w.r.t. the quasi-identifier attributes, and the tuples in each of the resulted clusters are generalized to a common tuple. To ensure that *k*-anonymity is correctly enforced, two constraints are required when the clustering process is performed. First, each resulted cluster must have at least *k* elements. If it does, the subsequent generalization of the cluster elements to a common tuple ensures that in the released microdata each tuple becomes indistinguishable from at least (*k*-1) other tuples. Second, the clustering method must act towards minimizing the information loss. The clusters should be formed such that the information lost by generalizing each group of tuples to a common value will be as low as possible. There are different ways to define information loss in the context of *k*-anonymization by clustering [1, 3].

Next, we formally introduce the *k*-anonymization by clustering problem, we explain how generalization of tuples in a cluster is performed, and we describe the information loss measure we use.

Definition 2: ([3]) Let $\mathcal{IM}$ be the initial microdata set. The ***k-anonymization by clustering problem*** is to find a partition $\mathcal{S} = \{cl_1, cl_2, \dots, cl_v\}$ of $\mathcal{IM}$, where $cl_j \subseteq \mathcal{IM}$, j=1..v, are called clusters and: $\bigcup_{j=1}^{v} cl_j = \mathcal{IM}$; $cl_i \bigcap cl_j = \varnothing$, i, j=1..v, $i \neq j$; $|cl_j| \geq k$, j=1..v ; $\sum_{j=1}^{v} InformationLoss(cl_j)$ is minimized (the *InformationLoss* measure for a cluster is introduced in Definition 4).

The *k*-anonymization of the initial microdata must be conducted to preserve data usefulness and minimize information loss. In order to achieve this goal, we generalize each cluster to the least general tuple that represents all tuples in that group. We call *generalization information* for a cluster the minimal covering tuple for that cluster, and we define it as follows.

Definition 3: Let $cl = \{r_1, r_2, \ldots, r_q\} \in \mathcal{S}$ be a cluster, $\mathcal{KN} = \{N_1, N_2, ..., N_s\}$ be the set of numerical quasi-identifier attributes and $\mathcal{KC} = \{C_1, C_2,,\ldots, C_t\}$ be the set of categorical quasi-identifier attributes. The ***generalization information of cl***, w.r.t. quasi-identifier attribute set $\mathcal{K} = \mathcal{KN} \cup \mathcal{KC}$ is the "tuple" *gen*(*cl*), having the scheme $\mathcal{K}$, where:

- For each categorical attribute $C_j \in \mathcal{K}$, $gen(cl)[C_j]$ = the lowest common ancestor in $\mathcal{H}_{Cj}$ of $\{r_1[C_j], r_2[C_j], \ldots, r_q[C_j]\}$;
- For each numerical attribute $N_j \in \mathcal{K}$, $gen(cl)[N_j]$ = the interval $[\min\{r_1[N_j], r_2[N_j], \ldots, r_q[N_j]\}, \max\{r_1[N_j], r_2[N_j], \ldots, r_q[N_j]\}]$.

For cluster *cl*, its generalization information *gen*(*cl*) is the tuple having as value for each quasi-identifier attribute, numerical or categorical, the most specific common generalized value for all that attribute values from *cl* tuples. In $\mathcal{MM}$, each tuple from cluster *cl* will be replaced by *gen*(*cl*).

To quantify the information loss caused by generalizing a cluster to a common tuple we use the measure introduced in [3].

Definition 4: Let $cl \in \mathcal{S}$ be a cluster, *gen*(*cl*) its generalization information and $\mathcal{K} = \{N_1, N_2, .., N_s, C_1, C_2, .., C_t\}$ the set of quasi-identifier attributes. The ***information loss*** caused by generalizing *cl* tuples to *gen*(*cl*) is:

$$IL(cl) = |cl| \cdot \left(\sum_{j=1}^{s} \frac{size(gen(cl)[N_j])}{size\left(\left[\min_{r \in \mathcal{IM}} r[N_j], \max_{r \in \mathcal{IM}} r[N_j] \right] \right)} + \sum_{j=1}^{t} \frac{height(\Lambda(gen(cl)[C_j]))}{height(H_{C_j})} \right)$$

where:

- $|cl|$ denotes the cluster *cl* cardinality;
- $size([i_1, i_2])$ is the size of the interval $[i_1, i_2]$ (the value i_2-i_1);
- $\Lambda(w)$, $w \in H_{Cj}$ is the subhierarchy of H_{Cj} rooted in w;
- $height(H_{Cj})$ denotes the height of the tree hierarchy H_{Cj}.

Definition 5 ([3]): ***Total information loss*** for a solution $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_v\}$ of the *k*-anonymization by clustering problem, denoted by $IL(\mathcal{S})$, is the sum of the information loss measure for all the clusters in $\mathcal{S}$.

3. THE INCREMENTAL MAINTENANCE OF MASKED MICRODATA ALGORITHM (*I3M* ALGORITHM)

As expressed by Definition 4, the information loss measure penalizes each tuple with a cost proportional with how "far" the tuple is from the cluster generalization information. Intuitively, smaller the clusters in a solution are and more similar the tuples in those groups will be, then less information will be lost. So, the desideratum is to group together the most similar objects (i.e. that cause the least possible generalization) in clusters with cardinalities as close as possible to *k*. A greedy algorithm for the *k*-anonymization by clustering problem can be found in [3].

We describe next the algorithms for the incremental maintenance of masked microdata $\mathcal{MM}$, when $\mathcal{IM}$ is modified by insert/ delete/update operations. Of course, *k*-anonymizing the modified microdata is possible by re-applying on the entire modified dataset the algorithm initially used for masking. But this process is time-consuming. The incremental algorithm we propose runs much faster, especially for small modification amounts, and it does not lose significantly in the quality of the solution it produces.

We present, first, the case when only inserts are being performed, then the case when only deletions are made. Updates can be managed as deletion of old tuples values followed by insertion of their new values (algorithm *I3M*-, then *I3M*+).

3.1 Maintaining $\mathcal{MM}$ when inserting

Let $\mathcal{S} = \{cl_1, cl_2,\ldots, cl_v\}$ be a solution for the *k*-anonymization by clustering problem, for the microdata $\mathcal{IM}$. The incremental dataset $\Delta^+\mathcal{M}$ is subsequently added to $\mathcal{IM}$. The problem is to efficiently update $\mathcal{S}$ to $\mathcal{S}' = \{cl'_1, cl'_2,\ldots, cl'_{v'}\}$ that ensures *k*-anonymity for $\mathcal{IM} \cup \Delta^+\mathcal{M}$.

The solution to this problem is straightforward. Each tuple *r* in $\Delta^+\mathcal{M}$ is added to that cluster in $\mathcal{S}$ that, increased with *r*, will produce the minimum increase of total information loss. The function *FindBestCluster***,** not explicitly described in Figure 1, identifies that optimal cluster for an tuple *r*, denoted by cl'_{u*}. When, due to multiple insertions, a cluster cl'_{u*} grows bigger than $2 \cdot k$ elements, that cluster will be split in two, not arbitrarily, but in a greedy manner that tries to minimize total information loss. The split procedure is described in function *Split2kCluster*, Figure 1. Figure 1 presents the *I3M*+ algorithm for maintaining *k*-anonymity for $\mathcal{MM}$ when new records are added.

Split2kCluster is inspired, though adapted, from hierarchical divisive clustering algorithms [7], for splitting the cluster cl'_{u*}. The tuple that reduces most the information loss for the cluster cl'_{u*} when deleted from cl'_{u*} is chosen as seed for a new cluster cl'_{new}. Tuples are transported one at a time from the cluster to be split to the new cluster until cl'_{new} has enough (*k*) tuples. The tuple to be moved between the two clusters is chosen to minimize the information loss cumulated for cl'_{u*} and cl'_{new}.

```
Algorithm I3M+ is
Input   IM – microdata; Δ+M – new microdata;
        S={cl1,cl2,…,clv} a solution for the k-
        anonymization by clustering problem for
        IM;
Output S'={cl'1,cl'2,…,cl'v'} a solution for the k-
        anonymization by clustering problem for IM
        ∪ Δ+M.
S' = S;
For every r ∈ Δ+M do
   cl'u* = FindBestCluster(r, S');
   cl'u* = cl'u* ∪ {r};
   If |cl'u*| ≥ 2·k then
      Split2kCluster(S', cl'u*);
   End If;
End For;
End I3M+.

Function Split2kCluster(S', cl'u*) is
   cl'new = ∅;  // a new empty cluster;
   While | cl'new | < k do
   /* Transfer from cluster cl'u* to cluster
      cl'new the tuple (among the tuples
      in cl'u*) that minimizes the sum of
      information loss for the two clusters */
      bestEntity = null;
      sumInfoLoss = ∞;
      For every r ∈ cl'u* do
         If IL(cl'u* –{r}) + IL(cl'new ∪{r})
             < sumInfoLoss then
            bestEntity = r;
            sumInfoLoss = IL(cl'u* –{r}) +
                          IL(cl'new ∪{r});
         End If;
      End For;
      cl'u* = cl'u* – {bestEntity};
      cl'new = cl'new ∪ {bestEntity}
   End While;
   S' = S' ∪ {cl'new};
End Split2kCluster;
```

Figure 1. The *I3M+* Algorithm

3.2 Maintaining $\mathcal{MM}$ when deleting

Let $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_v\}$ be a solution for the *k*-anonymization by clustering problem, for the microdata $\mathcal{IM}$. Obsolete tuples in dataset $\Delta^-\mathcal{M}$ are subsequently deleted from $\mathcal{IM}$. The problem is to efficiently update $\mathcal{S}$ to $\mathcal{S}' = \{cl'_1, cl'_2, \ldots, cl'_{v'}\}$ that ensures *k*-anonymity for $\mathcal{IM} - \Delta^-\mathcal{M}$.

The algorithm proceeds as follows. Each tuple *r* in $\Delta^-\mathcal{M}$ is deleted from the cluster currently containing it. The clusters that remain with less than *k* elements are dispersed into the others, in order to not lose *k*-anonymity for $\mathcal{IM} - \Delta^-\mathcal{M}$. The spreading procedure for a cluster cl'_j is described in function *DisperseLessKCluster* (see Figure 2). Each element *r* of cl'_j is relocated to another cluster, the one that, increased with *r*, will produce the minimum increase of the total information loss. If, in the spreading and relocation process for a cluster, other cluster grows bigger then $2 \cdot k$ elements, that cluster will be split in two. The same procedure *Split2kCluster* as in *I3M+* is used with that end in view. Figure 2 presents the *I3M–* algorithm for maintaining *k*-anonymity for $\mathcal{MM}$ when obsolete tuples are deleted.

```
Algorithm I3M- is
Input   IM – microdata; Δ-M – obsolete microdata;
        S={cl1,cl2,…,clv} a solution for the k-
        anonymization by clustering problem for
        IM;
Output S'={cl'1,cl'2,…,cl'v'} a solution for the k-
        anonymization by clustering problem for IM
        – Δ-M.
S' = S;
For every r ∈ Δ-M do
   // cl'owner denotes the cluster containing r
   cl'owner = cl'owner – {r};
End For;
For every cl'j ∈ S' do
   If |cl'j| < k then
      DisperseLessKCluster(S', cl'j);
   End If;
End For;
End I3M-.

Function DisperseLessKCluster(S', cl'j)
   S' = S' – {cl'j};
   For every r ∈ cl'j do
      cl'u* = FindBestCluster(r, S');
      cl'u* = cl'u* ∪ {r};
      If |cl'u*| ≥ 2·k then
         Split2kCluster(S', cl'u*);
      End If;
   End For;
End DisperseLessKCluster;
```

Figure 2. The *I3M–* Algorithm

4. THE OPTIMIZATION ALGORITHM

Both the non-incremental solution proposed in [3] for *k*-anonymization, and the incremental algorithm for maintaining *k*-anonymity for masked microdata when changes are brought to the initial microdata are based on greedy strategies for constructing solutions for the problem they address. So, they obtain good solutions, but not the optimal solution. We present next an optimization method, which is able to improve the solutions provided by the mentioned algorithm, w.r.t. the information loss measure. Although nor this method will obtain the optimal solution for the *k*-anonymity problem, it has the capacity to consistently and efficiently improve, to a certain degree, the solutions produced by the other two algorithms.

Intuitively, in a clustering solution for *k*-anonymization, there are some tuples that would be better placed in other clusters than the cluster currently containing them. This situation occurs because the total information loss will be smaller if some objects are moved to other clusters. The tuples re-placement can be of course, performed, only by preserving the *k*-anonymity for the microdata, i.e. no cluster can remain with less than *k* tuples. Our proposed solution treats both aspects: we perform certain elements relocations so that total information loss decreases, and every cluster ends with at least *k* elements. Specifically, we relocate not isolated elements, but entire clusters if this can be done and total information decreases.

In the following, we define the concepts we need for presenting our solution, and then we present the optimization algorithm, called *IL_Optimization* (Information Loss Optimization).

4.1 Totally Covered Clusters

In order to establish the conditions when entire clusters can be relocated and when cluster relocation is beneficial we introduce several definitions.

Definition 6: We say that a tuple ***r is covered by a cluster cl*** if and only if the following two conditions hold simultaneously:

- $r \notin cl$;
- If r is added to cl, $gen(cl)$ does not change.

Definition 7: Let $\mathcal{S} = \{cl_1, cl_2, \ldots, cl_v\}$ be a set of clusters. We call $cl \in \mathcal{S}$ ***totally covered*** (by $\mathcal{S}$) if and only if for all elements $r \in cl$ exists a cluster cl' in $\mathcal{S}$ distinct from cl such that r is covered by cl'. Two distinct elements in cl may be covered by different clusters.

Definition 8: We use the term ***break the cluster*** cl, when each element from the totally covered cluster cl is moved to a different cluster that covers it. This procedure is not unique since every element of cl may be covered by more than one cluster. To represent a particular break we use a triplet (cl_j, $\mathcal{S}$, $\mathcal{S}'$) (called a ***break***), where cl_j is a cluster from $\mathcal{S}$, and $\mathcal{S}'$ is the set of clusters obtained by breaking cl_j.

Definition 9: A break (cl_j, $\mathcal{S}$, $\mathcal{S}'$) is called ***interesting*** if the total information loss for the new set of clusters is less than the total information loss for the initial set.

In other words, for the new set of clusters $\mathcal{S}' = \{ cl_i^{'} \mid i = 1, .., v$ and $i \neq j\}$ where each $cl_i^{'}$ contains all elements from cl_i and it may also contain some elements from cl_j that are covered by cl_i, the following formula is satisfied:

$$\sum_{i=1}^{v} IL(cl_i) > \sum_{i=1,\, i\neq j}^{v} IL(cl_i^{'}) .$$

Definition 10: A break (cl_j, $\mathcal{S}$, $\mathcal{S}'$) is called ***optimal*** if $\mathcal{S}'$ has the lowest possible total information loss for all possible breaks of cluster cl_j.

Definition 11: Let $\mathcal{S} = \{cl_1, cl_2, .., cl_v\}$ be a set of clusters. We call $cl_j \in \mathcal{S}$ ***interesting totally covered cluster*** by $\mathcal{S}$ if and only if the following two conditions hold:

- cl_j is a totally covered cluster by $\mathcal{S}$;
- There is at least one break (cl_j, $\mathcal{S}$, $\mathcal{S}'$) that is interesting.

Property 1: If cl_j is an interesting totally covered cluster by $\mathcal{S}$ then every optimal break of cl_j is also an interesting break.

Proof: If cl_j is an interesting totally covered cluster by $\mathcal{S}$, then, according to Definition 11, it exists one interesting break (cl_j, $\mathcal{S}$, $\mathcal{S}'$), i.e. $IL(\mathcal{S}') < IL(\mathcal{S})$. For every optimal break (cl_j, $\mathcal{S}$, $\mathcal{S}''$), according to Definition 10, $IL(\mathcal{S}'') \leq IL(\mathcal{S}')$. From these two inequalities it follows that $IL(\mathcal{S}'') < IL(\mathcal{S})$, which means that the optimal break (cl_j, $\mathcal{S}$, $\mathcal{S}''$) is interesting.

Example: The tuples from Table 1 have three quasi-identifier attributes: *Age*, *ZipCode*, and *Gender*. The first is a numerical attribute, and the next two are categorical. For *ZipCode* attribute we use a generalization hierarchy that removes one last digit at the time for each level up the generalization tree, and for *Gender* attribute we have only one generalization level. The initial set of clusters, $\mathcal{S}$, contains three elements: cl_1, cl_2, and cl_3. The information loss for $\mathcal{S}$ is: $IL(\mathcal{S}) = 13.23$.

Table 1: Totally Covered Clusters Example

Clusters	Tuples	Age	ZipCode	Gender
cl_1	r_1	25	41076	Male
	r_2	40	41935	Female
cl_2	r_3	35	12345	Male
	r_4	55	33333	Male
cl_3	r_5	*33*	*41733*	*Female*
	r_6	*42*	*41076*	*Male*
	r_7	***38***	***41933***	***Male***

The cluster cl_3 is totally covered, each tuple being covered by one cluster (r_5 is covered by cl_1 and r_6 is covered by cl_2) or by both (r_7 is covered by cl_1 and cl_2). There are two possible breaks for cl_3. The sets of clusters following each break are:

$\mathcal{S}^1 = \{ cl_1^1 , cl_2^1 \}$, $cl_1^1 = \{r_1, r_2, r_5, r_7\}, cl_2^1 = \{r_3, r_4, r_6\}$ and

$\mathcal{S}^2 = \{ cl_1^2 , cl_2^2 \}$, $cl_1^2 = \{r_1, r_2, r_5\}, cl_2^2 = \{r_3, r_4, r_6, r_7\}$.

The information loss for the resulting sets of clusters is as follows: $IL(\mathcal{S}^1)$=13.4, $IL(\mathcal{S}^2)$=12.96. Compared with the original set of clusters, the information loss is lowered only by $\mathcal{S}^2$. Therefore, the break (cl_3, $\mathcal{S}$, $\mathcal{S}^2$) is an interesting break, while (cl_3, $\mathcal{S}$, = $\mathcal{S}^1$) is not.

4.2 The *IL_Optimization* Algorithm

When a greedy algorithm that solves *k*-anonymization problem by creating clusters is performed, usually we do not have any guarantee that all resulting clusters are not totally covered. We also notice that by breaking some of the totally covered clusters the information loss will decrease. Based on these facts we created an optimization algorithm that improves an existing clustering solution for the *k*-anonymization problem (Figure 3). The improvements resulted by applying the optimization algorithm are tested for both non-incremental algorithm introduced in [3] and the incremental algorithms we proposed in the previous sections.

5. EXPERIMENTAL RESULTS

In our experiments we used the *Adult* database from the UC Irvine Machine Learning Repository [16]. This database has become the benchmark for *k*-anonymity algorithms, being used by many researchers [11]. The experiments reported in [3] are also based on it, and in this section we compare, in terms of efficiency, scalability, and results quality, the static algorithm from [3] with our incremental algorithms.

The algorithms we tested have been implemented in Java, and tests were executed on a dual processor PC machine with 3.0 GHz each and 1 GB of RAM.

In all the experiments, we considered *Age* and *Education-num* as the set of numerical quasi-identifier attributes, and *Work-class*, *Marital-status*, *Occupation*, *Race*, *Sex*, and *Native-country* as the set of categorical quasi-identifier attributes. Microdata *k*-anonymity was enforced in respect to the quasi-identifier consisting of all these 8 attributes. We removed all tuples that contained the null (?) value for one or more of the quasi-identifier attributes from the microdata contained in the file adult.data [16].

Algorithm ***IL_Optimization*** is

Input $\mathcal{IM}$ – microdata;
$\mathcal{S}=\{cl_1, cl_2, \ldots, cl_v\}$ a solution for the k-anonymization by clustering problem for $\mathcal{IM}$;

Output $\mathcal{S}'=\{cl'_1, cl'_2, \ldots, cl'_{v'}\}$ another solution for the k-anonymization by clustering problem for $\mathcal{IM}$ so that $\sum_{j=1}^{v'} IL(cl'_j) \le \sum_{j=1}^{v} IL(cl_j)$.

Determine the set $\mathcal{S}_t = \{cl_{t1}, cl_{t2}, \ldots, cl_{tj}\}$ of totally covered clusters from $\mathcal{S}$ in descending order of their information loss. Let j be the number of totally covered clusters from $\mathcal{S}$ ($j = |\mathcal{S}_t|$)

$\mathcal{S}_{aux} = \mathcal{S}$;
$\mathcal{S}_f = \emptyset$;
For i =1 to j do
  ComputeOptimalBreak(cl_{ti}, $\mathcal{S}_{aux}$, $\mathcal{S}_f$);
  If (cl_{ti}, $\mathcal{S}_{aux}$, $\mathcal{S}_f$) is interesting then
    $\mathcal{S}_{aux} = \mathcal{S}_f$;
  End If;
End For;
$\mathcal{S}' = \mathcal{S}_{aux}$;

End *IL_Optimization*.
Function *ComputeOptimalBreak*(cl, $\mathcal{S}_{aux}$, $\mathcal{S}_f$) is
  $\mathcal{S}_f = \mathcal{S}_{aux}$;
  For every $r \in cl$ do
    Let $\mathcal{S}_{cover}$ be the set of all clusters from $\mathcal{S}_f$ that cover r.
    cl'= the cluster from $\mathcal{S}_{cover}$ with the lowest *IL* / tuple;
    $\mathcal{S}_f = \mathcal{S}_f - \{cl'\}$;
    $\mathcal{S}_f = \mathcal{S}_f \cup \{cl' \cup \{r\}\}$;
  End For;
End *ComputeOptimalBreak*;

Figure 3. The *IL_Optimization* Algorithm

A set of experiments has been conducted for a subset of the adult dataset, for testing both the *I3M+* and *I3M-* algorithms. Each experiment had three phases. First, the static algorithm from [3] was applied on a dataset $\mathcal{IM}$, which was a subset of the entire adult dataset. Second, we applied the *I3M+* / *I3M-* algorithm to update the clusters produced by the static algorithm, and considering several $\Delta^+\mathcal{M}$ / $\Delta^-\mathcal{M}$ datasets. Third, the static algorithm was applied on the entire new dataset, $\mathcal{IM} \cup \Delta^+\mathcal{M}$ / $\mathcal{IM} - \Delta^-\mathcal{M}$.

For *I3M+*, $\mathcal{IM}$ had 10000 objects, and $\Delta^+\mathcal{M}$ had different sizes, varying between 0.5% and 50% of $\mathcal{IM}$ size. For *I3M-*, $\mathcal{IM} - \Delta^-\mathcal{M}$ had 10000 objects, and $\Delta^-\mathcal{M}$ had different sizes, varying between 50 and 5000 tuples. The values considered for k were 3, 5 and 10.

The Figures 4 and 5 illustrate some of the obtained results, in terms of information loss and processing time. The reported results, for both static and incremental cases, are those obtained after the optimization algorithm was also applied.

In Figures 4.a and 5.a, we compare: a) the information loss for each set of clusters obtained by applying the static k-anonymization algorithm on the initial microdata $\mathcal{IM}$, followed by *I3M+* / *I3M–* on the corresponding increment dataset $\Delta^+\mathcal{M}$ / $\Delta^-\mathcal{M}$ with b) the information loss for the set of clusters obtained by applying the static k-anonymization algorithm on the final dataset $\mathcal{IM} \cup \Delta^+\mathcal{M}$ / $\mathcal{IM} - \Delta^-\mathcal{M}$. Of course, the information loss obtained by the incremental algorithm deteriorates when the increment/ decrement dataset grows in size w.r.t. the initial dataset size. Nevertheless, for small modification amounts, as is usually the case in the real world databases evolution, the information loss remains at about the same level as if we would use the non-incremental algorithm.

From these experiments, we draw the conclusion that the incremental algorithm can be used for updating k-anonymized microdata several times when the increment/decrement datasets, $\Delta^+\mathcal{M}$ / $\Delta^-\mathcal{M}$, are small. From time to time a static more time-consuming algorithm will be used to keep the information loss at acceptable ranges.

Figures 4.b and 5.b illustrate the running time for the incremental algorithm compared with the static algorithm. The time for incrementally processing the datasets $\Delta^+\mathcal{M}$ / $\Delta^-\mathcal{M}$ grows with datasets size, but it is significantly lower than the time required to statically process $\mathcal{IM} \cup \Delta^+\mathcal{M}$ / $\mathcal{IM} - \Delta^-\mathcal{M}$. The decision to apply the incremental algorithm vs. a static one is a tradeoff between data quality and processing time. However, the advantages are obvious: data quality remains high while the running time significantly decreases.

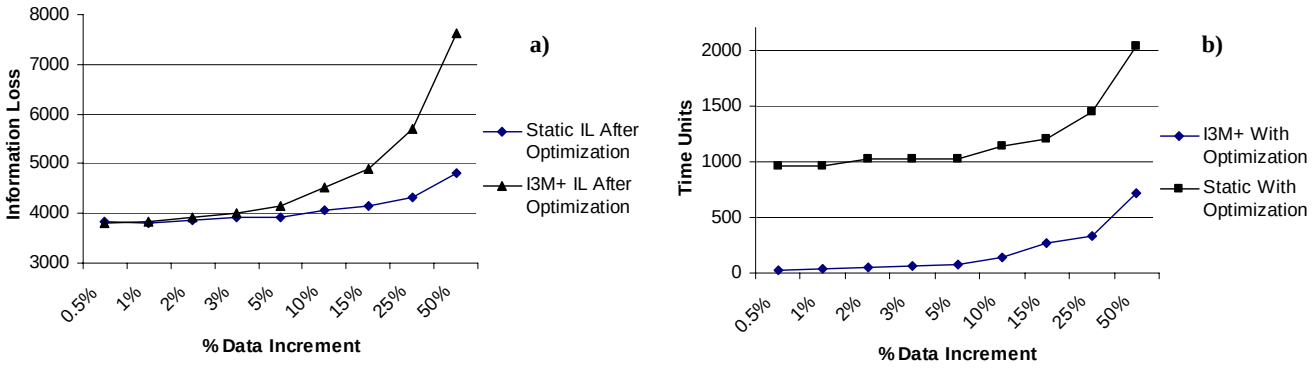

Figure 4. *IL* and Time for Static and *I3M+* Algorithms, k=3

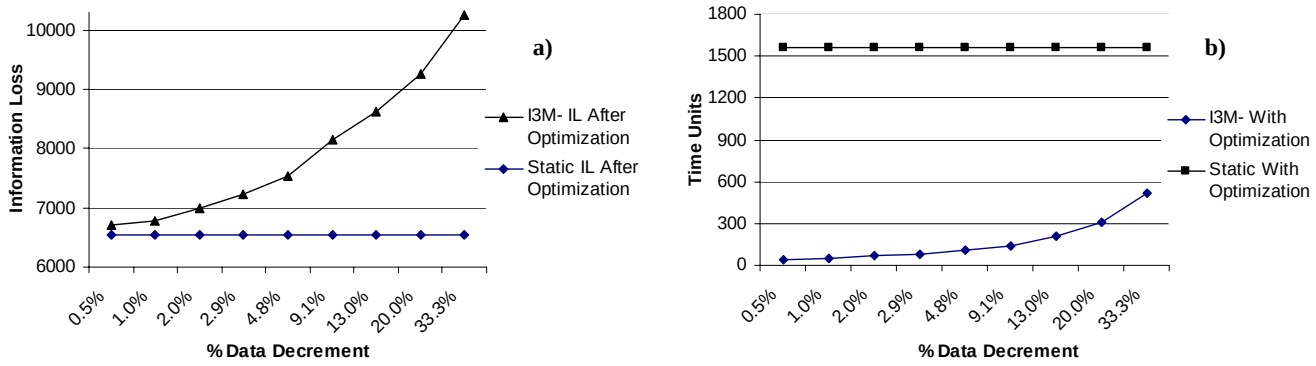

Figure 5. Static and *I3M*– Algorithms for *k*=5

As mentioned before, we applied, for both the static and the incremental algorithms, our *IL_Optimization* algorithm. In every experiment we obtained an improvement of the initial clustering solution w.r.t. information loss. Figure 6 graphically represents information loss before and after performing optimization, and processing vs optimization time, for experiments with static and *I3M+* algorithms. It can be seen that, for both static and incremental algorithms *IL_Optimization* has major advantages: it improves the prior solution and it requires a very small amount of time, compared to the time needed by the static or incremental *k*-anonymization algorithms**.**

6. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced an incremental updating technique for the maintenance of *k*-anonymized versions of large evolving datasets. Our experiments have shown that, for small increment/decrement datasets, $\Delta^{+}\mathcal{M}$ / $\Delta^{-}\mathcal{M}$, the running time of the incremental algorithms is significantly lower then the running time for the static algorithm. From the data quality perspective, the information loss computed using our incremental algorithms is comparable with the information loss obtained by applying the non-incremental algorithm to the final dataset. This technique will allow a data holder to make his data available in real time when small updates are performed over the initial dataset.

The second main contribution of this paper is an optimization algorithm that is able to improve the solutions attained by either the non-incremental or incremental algorithms. The optimization process reduces the information loss value while its execution time does not add a significant amount of time to the entire *k*-anonymization process.

There are several promising areas for future work. We plan to extend the incremental updating and optimization techniques for *p*-sensitive *k*-anonymity property [21] also known as *l*-diversity [13]. We intend to explore to what extent the release of incremental k-anonymized versions of a masked microdata may lead to identity disclosure. We also analyze the possibility of using other clustering algorithms for *k*-anonymity problems that will perform better in terms of information loss and/or running time.

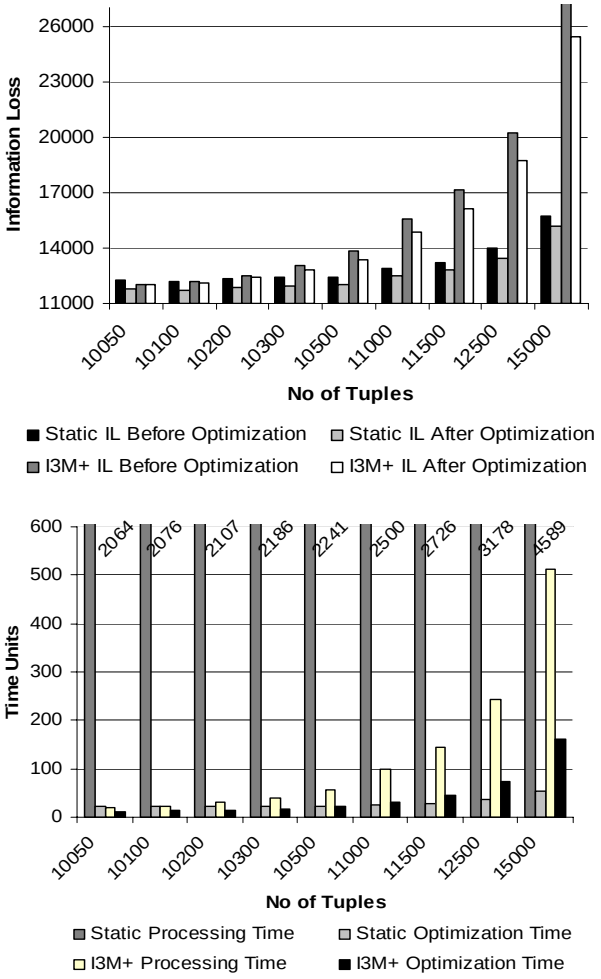

Figure 6. *IL* and Time for *IL_Optimization* Algorithm

7. REFERENCES

[1] Aggarwal, G., Feder, T., Kenthapadi, K., Motwani, R., Panigrahy, R., Thomas, D., and Zhu, A., Achieving Anonymity via Clustering, *Proc. of the 10th Intl. Conf. on Database Theory*, 2005.

[2] Bayardo, R.J, and Agrawal, R., Data Privacy through Optimal *k*-Anonymization, *Proc. of the IEEE Intl. Conf. of Data Eng.*, 2005, 217-228.

[3] Byun, J.W., Kamra, A., Bertino, E, and Li, N., Efficient *k*-Anonymity using Clustering Technique, *CERIAS Tech Report 2006-10*, 2006a.

[4] Byun, J.W., Sohn, Y., Bertino, E., and Li, N., Secure Anonymization for Incremental Datasets, *Proc. of the 3rd VLDB Workshop on Secure Data Management*, 2006b.

[5] Domingo-Ferrer, J., Mateo-Sanz, J., and Torra,V., Comparing SDC Methods for Microdata on the Basis of Information Loss and Disclosure Risk, *Pre-proc. of ETK-NTTS'2001 (vol. 2), Luxembourg: Eurostat*, 2001, 807-826.

[6] GLB, Gramm-Leach-Bliley Financial Services Modernization Act, available online at http://banking.senate.gov/conf/, 1999.

[7] Han, J., and Kamber, M., *Data Mining: Concepts and Techniques*, The Morgan Kaufmann Series in Data Management Systems, 2001.

[8] HIPAA, Health Insurance Portability and Accountability Act, Available online at http://www.hhs.gov/ocr/hipaa, 2002.

[9] Iyengar, V., Transforming Data to Satisfy Privacy Constraints, *Proc. of the Eighth ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, 279-288.

[10] LeFevre, K., DeWitt, D., and Ramakrishnan, R., Incognito: Efficient Full-Domain *k*-Anonymity, *Proc. of the ACM SIGMOD*, Baltimore, Maryland, 2005, 49-60.

[11] LeFevre, K., DeWitt, D., and Ramakrishnan, R., Mondrian Multidimensional *k*-Anonymity, *Proc. of the IEEE Intl. Conf. of Data Eng.*, Atlanta, 2006.

[12] Lunacek, M., Whitley, D., and Ray, I., A Crossover Operator for the *k*-Anonymity Problem, *Proc. of the GECCO Conference*, 2006, 1713 – 1720.

[13] Machanavajjhala, A., Gehrke, J., and Kifer, D., *l*-diversity: privacy beyond *k*-anonymity, *Proc. of the 22nd IEEE Intl. Conference on Data Eng.*, 2006.

[14] Mateo-Sanz, J.M., Domingo-Ferrer, J., and Sebe, F., Probabilistic Information Loss Measures in Confidentiality Protection of Continuous Microdata, *Data Mining and Knowledge Discovery*, Vol. 11, No. 2, 2005, 181– 193.

[15] Meyerson A., and Williams R., On the Complexity of Optimal *k*-Anonymity, *ACM PODS Conf.*, 2004, 223 – 228.

[16] Newman, D.J., Hettich, S., Blake, C.L., and Merz, C.J., UCI Repository of Machine Learning Databases, available at www.ics.uci.edu/~mlearn/MLRepository.html, University of California, Irvine, 1998.

[17] Samarati P., Protecting Respondents Identities in Microdata Release, *IEEE Transactions on Knowledge and Data Eng.*, Vol. 13, No. 6, 2001, 1010-1027.

[18] Sweeney L., *k*-Anonymity: A Model for Protecting Privacy, *Intl. Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, Vol. 10, No. 5, 2002a, 557 – 570.

[19] Sweeney L., Achieving *k*-Anonymity Privacy Protection Using Generalization and Suppression, *Intl. Journal on Uncertainty, Fuzziness, and Knowledge-based Systems*, Vol. 10, No. 5, 2002b, 571 – 588.

[20] Truta, T.M., Fotouhi, F., and Barth-Jones, D., Privacy and Confidentiality Management for the Microaggregation Disclosure Control Method, *Workshop on Privacy and Electronic Society, 10th ACM CCS*, 2003, 21-30.

[21] Truta, T.M., and Bindu, V., Privacy Protection: p-Sensitive k-Anonymity Property, *Workshop on Privacy Data Management, 22th IEEE Intl. Conf. of Data Eng.*, 2006.

[22] Willemborg, L., and Waal, T. (ed) , *Elements of Statistical Disclosure Control*, Springer Verlag, 2001.