Trading Privacy for Information Loss in the Blink of an Eye

Alexandra Pilalidou[1⋆] and Panos Vassiliadis[2]

[1] FMT Worldwide, Limassol,Cyprus
apilalid@gmail.com
[2] Dept. of Computer Science, Univ. of Ioannina, Hellas
pvassil@cs.uoi.gr

Abstract. The publishing of data with privacy guarantees is a task typically performed by a data curator who is expected to provide guarantees for the data he publishes in quantitative fashion, via a privacy criterion (e.g., k-anonymity, l-diversity). The anonymization of data is typically performed off-line. In this paper, we provide algorithmic tools that facilitate the negotiation for the anonymization scheme of a data set in user time. Our method takes as input a set of user constraints for (i) suppression, (ii) generalization and (iii) a privacy criterion (k-anonymity, l-diversity) and returns (a) either an anonymization scheme that fulfils these constraints or, (b) three approximations to the user request based on the idea of keeping the two of the three values of the user input fixed and finding the closest possible approximation for the third parameter. The proposed algorithm involves precomputing suitable histograms for all the different anonymization schemes that a global recoding method can follow. This allows computing exact answers extremely fast (in the order of few milliseconds).

1 Introduction

The area of privacy-preserving data publishing serves the purpose of allowing a data curator publish data that contain sensitive information for persons in the real world while serving the following two antagonistic goals: (a) allow well-meaning data miners extract useful knowledge from the data, and, (b) prevent attackers from linking the published, anonymized records to the individuals to which they refer. Frequently, the method of choice for the anonymization method involves the *generalization* of the values of the published tuples to values that are more abstract. This creates the possibility to *hide tuples in the crowd of similar tuples*. Typically, the privacy guarantee per tuple is expressed as a *privacy criterion*, e.g., k-anonymity or l-diversity ([1], [2]) that quantitatively assesses each of the groups of similar tuples (k-anonymity for its size, l-diversity also for the variance of sensitive values within the group). Overall, *the data curator has to negotiate antagonistic goals: (a) the request to avoid exceeding a certain threshold of deleted (suppressed) tuples,*

⋆ Work performed while in the Univ. of Ioannina

(b) maximum tolerated generalization heights per attributes that are acceptable by the end users, and, (c) the curator's constraint on the privacy value. Of course, it might not be possible to achieve a consensus on the parameters of the problem. So, the desideratum is that the curator interactively explores different anonymization possibilities. If, for example, the curator sets a suppression threshold too low for the data set to sustain while respecting the rest of the user criteria, then the system should ideally respond very quickly with a negative answer, along with a set of proposals on what possible generalizations, close to the one that he originally submitted, are attainable with the specific data set. As opposed to the state of the art methods that operate off-line, *our method informs the curator (a) in user time (i.e., ideally with no delay that a person can sense), (b) with the best possible solution that respects all the constraints posed by the involved stakeholders, if such a solution exists, or, (c) convenient suggestions that are close to the original desiderata around generalization, suppression and privacy.*

The research community has focused on different directions, complementary but insufficient for the problem. We refer the interested reader to the excellent survey of Fung et al. [3] for further probing on the state of the art. The most related works to our problem are (a) [4] who deal with the problem of finding the local recoding with the least suppressed cells (not tuples) without any hierarchies in mind, and, (b) [5], where the author works in a similar setting and prove that the probability of achieving k-anonymity tends to zero as the number of dimensions rises to infinite; the theoretical analysis is accompanied with a study of generated data sets (one of which rises up to 50 dimensions) that supports the theoretical claim. Still, compared to these works, our method is the first to simultaneously combine a focus to on-line response, generalization hierarchies, different values of k or l, and different privacy criteria.

Our approach is based on a method that involves precomputing statistical information for several possible generalization schemes. A *generalization scheme* is determined by deciding the level of generalization for every quasi-identifier – in other words, a generalization scheme is a vector characterizing every quasi-identifier with its level of generalization. To efficiently compute the amount of suppression for a given pair of (i) value for the privacy criterion, and, (ii) a generalization scheme, we resort to the *precalculation of a histogram* per generalization scheme that allows us to calculate the necessary statistical information. For example, in the case of k-anonymity, we group the data by the quasi identifier attribute set in their generalized form and we count how many groups have size 1, 2, ... etc. So, given a specific value of k, we can compute how many tuples will be suppressed for any generalization scheme. Similarly, in the case of naive l-diversity, for each group we count the number of different sensitive values and the size of the group too.

We organize generalization schemes as nodes in a lattice. A node v is lower than a node u in the lattice if u has at least one level of generalization higher than v for a certain quasi-identifier and the rest of the quasi-identifiers in higher or equal levels. The main algorithm exploits the histograms of the nodes in the lattice and checks whether there exists a node of height less than $\mathbf{h}$ that satisfies k (or, l) without suppressing

more than $MaxSupp$ tuples. This is performed by first checking the *top-acceptable-node* v_{max} defined with generalization levels $\mathbf{h}=[h_1^c, \ldots, h_n^c]$. If a solution exists, then we exploit a monotonicity property of the lattice and look for possible answers in quasi identifiers with less or equal generalization levels than the ones of the top acceptable node. In the case that no solution exists in the top acceptable node, the algorithm provides the user with 3 complementary suggestions as answers:

- The first suggested alternative satisfies k (or, l) and $\mathbf{h}$ but not *MaxSupp*. In fact, we search the space under the top acceptable node and provide the solution with the minimum number of suppressed tuples. In typical situations, we prove that the answer is already found in the top acceptable node and thus, provide the answer immediately.
- The second alternative is a solution that satisfies k (or, l) and *MaxSupp* but violates $\mathbf{h}$. This means that we have to explore the space of quasi identifiers that are found in generalization levels higher or equal than the top acceptable node. We exploit the lattice's monotonicity properties to avoid unnecessary checks and utilize a binary search exploration of heights on the lattice.
- The third suggested alternative is a solution that finds the maximum possible k (or, l) for which $\mathbf{h}$ and *MaxSupp* are respected for the quasi identifiers of the top acceptable node. Similarly to the first case, this answer can be provided immediately at the top acceptable node.

In Fig. 1 we depict the lattice for the Adult data set where we constrain the quasi identifier set to *Age*, $Work\ class$ and *Education*. Each node of the lattice is a generalization scheme; underneath each node you can see the number of tuples that violate the constraint of 3-anonymity. The label of each node shows the height of the generalization for each of the QI attributes. We pose two queries to the lattice, both constraining k to 3 and the height to level 1 for age, level 2 for $Work\ class$ and level 1 for *Education* (coded as 121 for short). Then, the *top-acceptable-node* v_{max} is 121 and the lattice it induces is depicted as the blue diamond over the lattice. We hide nodes 030 and 102 from the figure as they are not members of the sublattice induced by v_{max}. The first query involves a tolerance for 20 suppressed tuples; in this case, v_{max} suppresses less than 20 tuples and we know that its sub-lattice will produce an exact answer. Out of all candidates, node 111 provides the exact answer with the lowest height (and since we have a tie in terms of height with node 120, the one with the least suppression among the two). The second query requires a maximum suppression of 8 tuples; in this case, since v_{max} fails the constraint, we provide two suggestions concerning the relaxation of suppression and privacy directly at v_{max}, and, a third suggestion for the relaxation of the height constraint by exploring the full lattice (and ultimately resulting in node 400).

2 Anonymity Negotiation over a Full Lattice

In this section, we present Algorithm *SimpleAnonymityNegotiation*. The proposed algorithm takes as input a relation R to be generalized, a set

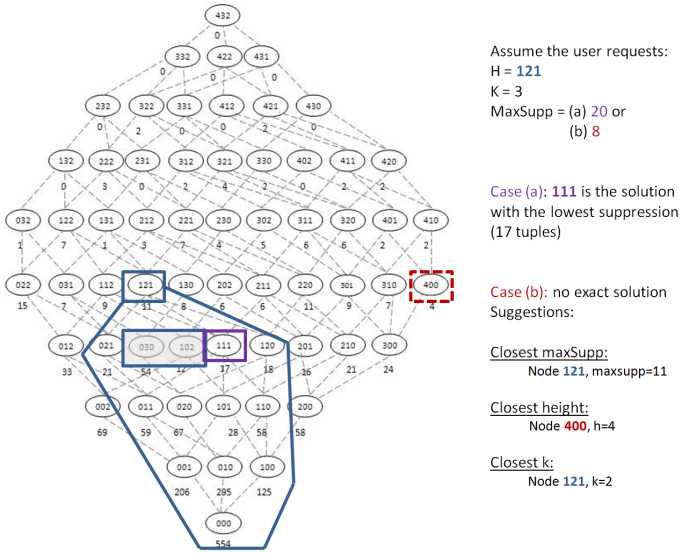

Fig. 1. Example of lattice and query answers. The lattice is annotated with suppressed tuples for |QI|= 3 and k=3. The QI is Age, Work class, Race.

of hierarchies **H** for the quasi-identifier attributes, the histogram lattice L for all possible combinations of the generalization levels, and the requirements for the maximum desirable generalization level per quasi-identifier (**h**), the maximum tolerable number of tuples to be suppressed (*maxSupp*) and the least size of a group (k or l), as the privacy constraint. The outputs of the algorithm are (a) either a node of the lattice (i.e., a generalization scheme) that provides the best possible *exact* solution to the user requirements (with best possible being interpreted as the one with the *lowest height*, and, if more than one candidate solutions have this lowest height, the one with the minimum suppression), or, (b) three suggestions for approximate answers to the user request, the first relaxing the number of suppressed tuples, the second relaxing the constraints on the heights per dimension and the third relaxing the minimum acceptable privacy criterion (e.g., k in k-anonymity).

Algorithm *SimpleAnonymityNegotiation* starts by identifying a reference node in the lattice, to which we refer a v_{max}. The node v_{max} is the node that satisfies all the constraints of **h** for the quasi-identifiers, at the topmost level; in other words, v_{max} is the highest possible node that can obtain an exact answer to the user's request. We will also refer to v_{max} as the *top-acceptable* node. Then, two cases can hold: (a) v_{max} is able to provide an exact solution (the if part), or (b) it is not, and thus we have to resort to approximate suggestions to the user (the *else* part). The check on whether a node can provide an exact solution is given by function *checkExactSolution* that looks up the histogram of a node v and performs the appropriate check depending on the privacy criterion.

```
Algorithm SimpleAnonymityNegotiation(L,k,h,MaxSupp)
In: Lattice L with the histograms for R,H, constraints for k, h, MaxSupp
Out: an exact solution s[v,k,h,supp] or s1,s2,s3, si=[v_i,k_i,h_i,supp_i]
Var: a 2D vector of candidate solutions Candidates[hmax][]
Begin
Let v_max be the node that corresponds to the constraint h ;
if v_max is visited then exit;
mark v_max as visited;
if (checkExactSolution(v_max,L,k,h,MaxSupp) == true){
        Candidates[height(v_max)] = Candidates[height(v_max)] ∪ {v_max};
        for all v_c in lower(v_max)

        ExactSublatticeSearch(v_c,L,k,h,MaxSupp,Candidates);
//when the recursion is over, the Candidates has the full list of nodes
//that can serve as candidate solutions
        minHeight = minimum height having Candidates[minHeight] != {};
        v_win = v in Candidates[minHeight] with the lowest possible suppression for k;
        return(v_win,k,minHeight,suppressed(v_win,k));
}
else{
        approxSol_1 = ApproximateMaxSupp(L,v_max,k,h,MaxSupp);
        approxSol_2=ApproximateH(L,v_max,height(v_max),height(top),k,h,MaxSupp);
        approxSol_3 = ApproximateK(L,v_max,k,h,MaxSupp);

        return approxSol_1, approxSol_2, approxSol_3;
}
End.
```

```
ExactSublatticeSearch(v,L,k,h,MaxSupp,Candidates){
        if v is visited then exit;
        mark v as visited;
        if (checkExactSolution(v,L,k,h,MaxSupp) == true){
                Candidates[height(v)] = Candidates[height(v)] U {v};
                for all v_c in lower(v)
                        ExactSublatticeSearch(v_c,L,k,h,MaxSupp,Candidates);
        }
}
```

```
checkExactSolution(v,L,k,h,MaxSupp){
        lookup histogram of v in L;
        if suppressed(v,k) <= MaxSupp && height(v) <= h  return true;
        else return false;
}
```

Fig. 2. Algorithm Simple Anonymity Negotiation and accompanying functions

Exact Answer. When an exact answer can be provided by the top-acceptable node v_{max}, then we can be sure that the sublattice induced by v_{max} contains an exact answer; however, we need to discover the one with the minimum possible height and, therefore, we need to descend down the lattice to discover it. The auxiliary variable *Candidates* holds all the nodes that conform to the user request, organized per height. Each time such a node is found, it is added to *Candidates* at the appropriate level (Line 5) and its descendants (returned via the function *lower*()) are recursively explored via the call of function *ExactSublatticeSearch*. When the lattice is appropriately explored we need to find the lowest level with a solution in the lattice, and, among the (several candidate) solutions of this level we must pick the one with the least suppression.

If node v_{max} fails to provide an answer that conforms to the user request, then we are certain that it is impossible to derive such a conforming answer from our lattice and we need to search for approximations. So, we provide the users with suggestions on the possible relaxations that can be made to his criteria.

```
ApproximateMaxSupp(L,v,k,h,MaxSupp){
        find the minimum amount of suppressed tuples, approxSupp, s.t.
checkExactSolution(v,L,k,h,approxSupp) returns true;
        if no such value exists, return {};
        else{
              for all v_c in sublattice(v) (recursively){
                    checkExactSolution(v_c,L,k,h,approxSupp)
                      break when a whole level fails to produce a solution;
              }
             let v_win be the node with the lowest height that satisfies k,h,approxSupp
             (with arbitrary tie resolution)
        return v_win,k,h,approxSupp;
        }
}
```

```
ApproximateH(L,v,h_low,h_high,k,h,MaxSupp){
    while(h_low <= h_high){
        h_current = middle between h_low and h_high;
        flag = checkIfNoSolutionInCurrentHeight(L,h_current,k,MaxSupp);
        if (flag == true){
            low = current + 1;
        }
        else{
            currentMinHeight = current;
            high = current - 1;
         }
    }
    for all v_c in currentMinHeight, find the one v_win, with the minimum suppressed(v_c,k);
        //exception: this fails only if k > |R|, else top of the lattice always answers
        return v_win,k,height(v_win),MaxSupp;
}


checkIfNoSolutionInCurrentHeight(L,h_current,k,MaxSupp){
        for all v_c in h_current
        if suppressed(v_c,k) <= MaxSupp return false;
        return true;
}
```

Fig. 3. Approximation Functions

Suppression and privacy relaxations. Function *ApproximateMaxSupp* respects the privacy criterion k and the max tolerable height **h**, and returns the best possible relaxation with respect to the number of suppressed tuples. Since **h** is to be respected, we are restricted in the sub-lattice induced by v_{max}. Since v_{max} has failed to provide a conforming answer, no node in the sublattice can provide such an answer, either. So, we assess the number of tuples that have to be suppressed if we retain k fixed and stay at the highest candidate node v_{max}. Observe that any node in the sublattice of v_{max} will result in higher or equal number of suppressed tuples – remember that the lower we go, the smaller the groups are and the higher the suppression. In other words, it will either be v_{max} that will give the answer or one of its descendants in the rare case that the groups of the descendant are mapped one to one to the groups of v_{max}, thus resulting in exactly the same number of suppressed tuples. The relaxation of privacy is identical (omitted for lack of space).

Height relaxation. Function *ApproximateH* retains the maximum tolerable number of suppressed tuples *MaxSupp* and the privacy criterion of k and tries to determine what is the lowest height h that can provide an answer for these constraints. This time, we operate outside the borders

of the sublattice of v_{max} since **h** is not to be respected. The function *ApproximateH* performs a binary search on the height between the height of v_{max} and the upper possible height (the top of the lattice). Every time a level is chosen, we start to check its nodes for possible solutions via the function *checkIfNoSolutionInCurrentHeight*. If the function explores a height fully and fails to find an answer, this is an indication that we should not search lower than this height (remember: failure to find a solution signals for ascending in the lattice). Every time the function finds a node that can answer, then we must search in the lower heights for possibly lower solutions. When the binary search stops, the value *currentMinHeight* signifies the lowest possible height where a solution is found. Then, we explore this height fully to determine the node with the minimum suppression.

3 Experiments

We present our results over the Adult data set [6] over two privacy criteria: k-anonymity and naive l-diversity. We have assessed the performance in terms of time and visited nodes as we vary the value for the privacy criterion, the maximum tolerable generalization height and the maximum tolerable amount of suppressed tuples. We performed 28 user requests: each time we keep the two out of the three parameters fixed in their middle value and vary QI with 3, 4, 5, 6 attributes as well as the parameters under investigation. The values of k range in 3, 10, 50. The values for l are smaller (3,6,9) in order to avoid suppressing the entire data set. The values for **h** are: (a) low heights having levels with heights 1 and 0, (b) middle heights having mostly 2 and few 1 level heights and, (c) middle-low in between (remember that they vary per QI). In all our experiments we have used a Core Duo 2.5GHz server with 3GB of memory and 300GB hard disk, Ubuntu 8.10, and MySQL 5.0.67. The code is written in Java.

Effectiveness and Efficiency. The *increase of the privacy criterion* (Fig. 14, k) has divergent effects. When QI is small, there is an exact answer and the search is directed towards lower heights. Consequently, as k increases the solution is found earlier. On the contrary, for larger QI sizes and relaxations to user request, the increase of k sublinearly increases the search space.

The *increase of the maximum tolerable height* (Fig 14, **h**) has a consistent behavior. When the QI size is small, we can have exact solutions; in this case, when we increase the maximum tolerable height, this increases the search space too. In contrast, when relaxations are sought, the higher the constraint, the faster a solution is found.

The *constraint on the maximum tolerable suppression* (Fig. 14, $MaxSupp$) is similar: the higher the constraint is set, the faster an approximate solution is found (except for low QI sizes where exact answers are possible and the behavior is inverse due to the exploration of $L(v_{max})$).

In all experiments, it is clear that the costs are dominated by the QI size. *Finally, in all experiments, the times ranged between 1 and 8 msec, thus facilitating the online negotiation of privacy with the user, in user*

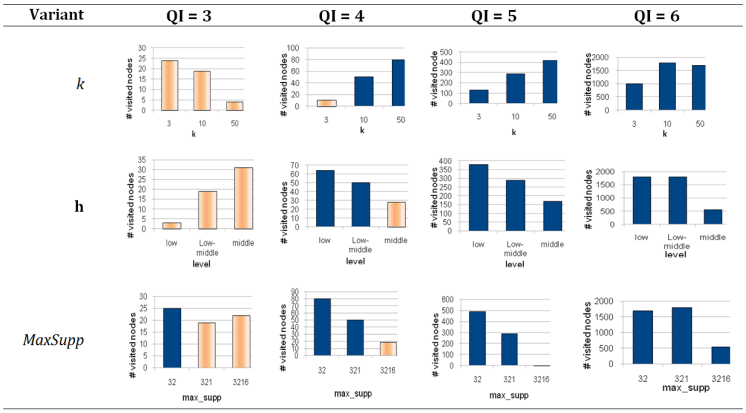

Fig. 4. Number of visited nodes for different QI, k, h, MaxSupp. *All times range between 1 and 8 msec.* Light coloring is for exact matches and dark coloring is for approximate matches

time. The experiments with l-diversity demonstrate a similar behavior as the experiments of k-anonymity. See [7] for a detailed report of all our experiments.

The price of histograms. The lattice of generalization schemes and most importantly, the histograms with which the lattice is annotated come with a price, both in terms of space and in terms of construction time. The lattice construction is negligible in terms of time; however, this does not hold for its histograms: observe that as the QI size increases, the time for the histogram construction increases exponentially (Fig. 5). The reason for this phenomenon is depicted in Fig. 5 where the number of nodes per QI size is also depicted. Although the time spent to construct the histograms is significant, the amount of memory that is needed to keep the histograms in main memory is quite small (e.g., we need approx. 3 MB for the largest QI size for k-anonymity; the value drops to approx. 1 MB for l-diversity, since the number of discrete values that the histograms take are much lower in this case).

QI size	No. nodes	Avg constr. time for k-anon (min)	Avg constr. time for l-div (min)
QI=3	60	0,141	0,1858
QI=4	180	0,602	0,702
QI=5	900	3,7	4,53
QI=6	3600	19,02	21,12

Fig. 5. Lattice size in no. nodes (left); average construction time (min.) for the full lattice and the respective histograms for k-anonymity (middle) and l-diversity (right) over the Adult data set

4 Summary and Pointers for Further Probing

In this paper, we report on our method that allows a data curator trade information loss (expressed as tuple suppression and increase in generalization levels) for privacy (expressed as the value of a privacy criterion like k-anonymity or naive l-diversity). *The full version of this paper [7] includes material that is omitted here for lack of space.* Specifically:

- We theoretically prove that *our method is guaranteed to provide the best possible answers* for the given user requests.
- We provide an extensive discussion on the validity of the problem. To the best of our knowledge, [7] is the first to perform a systematic study and report on the interdependency of suppression, generalization and privacy in a quantitative fashion.
- We provide extensive experimental results, in full detail for all the different combinations of QI size, k or l. Moreover, all the experiments have also been performed on the IPUMS data set, and the reported results demonstrate a similar behavior with the Adult data set.
- To handle the issue of scale (as the off-line lattice-and-histogram construction is dominated by both the QI size and the data size) we provide a method for the selection of a small subset of characteristic nodes of the lattice to be annotated with histograms, based on a small number of tests that rank QI levels for the grouping power.

Acknowledgments. We would like to thank X. Xiao and Y. Tao for explanations concerning the IPUMS data set.

References

1. Samarati, P.: Protecting respondents' identities in microdata release. IEEE Trans. Knowl. Data Eng. **13**(6) (2001) 1010–1027
2. LeFevre, K., DeWitt, D.J., Ramakrishnan, R.: Incognito: Efficient full-domain k-anonymity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005. (2005) 49–60
3. Fung, B.C.M., Wang, K., Chen, R., Yu, P.S.: Privacy-preserving data publishing: A survey of recent developments. ACM Comput. Surv. **42**(4) (2010)
4. Park, H., Shim, K.: Approximate algorithms for k-anonymity. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007. (2007) 67–78
5. Aggarwal, C.C.: On k-anonymity and the curse of dimensionality. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, August 30 - September 2, 2005. (2005) 901–909
6. U.C. Irvine Repository of Machine Learning Databases: Adult data set. Available at http://www.ics.uci.edu/ mlearn (1998)
7. Pilalidou, A.: On-line negotiation for privacy preserving data publishing. (MSc Thesis. MT 2010-15, Dept. of Computer Science, Univ. of Ioannina, 2010. Also available at: http://www.cs.uoi.gr/∼pvassil/publications/2012_SSDBM/)