ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

PATIENT DATA SET PUBLICATION THROUGH DIFFERENTIAL PRIVACY VIA WAVELET TRANSFORMS

Mr. P.M.GAVALI[1], Prof. P.C.BHASKAR[2]

Computer Science and Technology, Shivaji University, Kolhapur, Maharashtra, India[1]

Computer Science and Technology, Shivaji University, Kolhapur, Maharashtra, India[2]

Abstract: Every organization stores a required data in the digital form. This data needs to be published due to the mutual benefits or due to the government rule. This data also has significant research value. While publishing a data in its original form, there is loss of privacy of the individual record. To avoid this, we can use ε -differential privacy. Dwork provides a simplest way to achieve this. But it does not work well with range count query. In this paper we have used Privelet+ method using array based implementation for wavelet transforms and Privelet* method. These methods show better result than the Dwork method for the range count queries.

Keywords Data Publication, Wavelet Transforms

I. INTRODUCTION

Today every organization stores a required data in the digital form. For example, hospital stores data related with patients in digital form. This data needs to be published either due to mutual benefits or due to the government rule. For example, every licensed hospital in California is required to submit data related with the patient when patient gets discharge from the hospital [Carlisle, *et al.* (2007)]. This data includes sensitive information related with individual. If we publish this data in its original form, it violets the privacy of individual. Methods that deal with this problem are known as Privacy Preserving Data Publishing Techniques [Fung B.C.M, *et al.* (2010)]. These techniques publish data in such a way that published data remains practically useful while individual privacy is preserved. One of the methods for privacy preserving data publishing technique is ε -differential privacy [Dwork C., *et. al.* (2006)]. This method says, a randomized algorithm G satisfies ε-differential privacy, if 1) for any two tables T_1 and T_2 that differ only in one tuple, and 2) for any output O of G, we have $Pr\{G(T_1) =O\} \leq e^{\varepsilon} Pr\{G(T_2)= O\}$

Simplest method to achieve a ε-differential privacy is provided by Dwork [Dwork C., *et. al.* (2006)]. This method first computes the frequency distribution of the tuples in the input data and then publish a noisy version of the distribution by adding the noise of θ(1) variance to each entry. The noisy frequency matrix preserves privacy, as it conceals the exact data distribution. In addition, the matrix can provide approximate results for all range queries. But this method fails to provide useful results. In particular, for a count query answered by taking the sum of a constant fraction of the entries in the noisy frequency matrix, the approximate query result has a θ(m) noise variance, where m denotes the total number of entries in the matrix and m is typically an enormous number, as practical data sets often contain multiple attributes with large domains.

To deal with this problem we have implemented Privelet+ method [Xiaokui Xiao, *et al.* (2011)] and Privelet* method [Xiaokui Xiao, *et al.* (2011)]. But while implementation we have used array based implementation for wavelet transforms rather than tree based implementation. This method provides more accurate range count query answers than the Dwork's method.

II. PROBLEM STATEMENT

Suppose we want to publish a table that T with $A_1,A_2,..,A_n$ domains. In this, we try to optimize the range count query of the form

SELECT COUNT (*) FROM T

WHERE $A_1 \in S_1$ AND $A_2 \in S_2$ AND,..,$A_n \in S_n$

$S_1,S_2..S_n$ are intervals defined on domain $A_1,A_2...A_n$

III. PROPOSED STRUCTURE

A. Priv*elet+ method*

Privelet+ technique takes three inputs from the user. First input is table that we want to publish. Second is error magnitude that we want to add in published data and third is attribute list for further processing. This algorithm works in following different steps:

Step 1) Calculating frequency matrix:

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

For the input table, we need to calculate the frequency matrix on one of the attribute. It computes the frequency distribution of the tuples in the input data. For example, suppose we want to publish following patient table having attributes as Patient Name, Patient Address, Patient Job, Patient Status and Patient Age. Patient Status specifies whether the person is suffering from diabetes or not.

Table I. Medical Data

Patient Name	Patient Address	Patient Job	Patient Status	Patient Age
Yogesh Laga	Pune	Teacher	Yes	41
Arnika Shetti	Pune	Doctor	Yes	42
Dhiraj Kumar	Pune	Doctor	No	43
Amol Teke	Pune	Teacher	No	41
Sandip Todakar	Pune	Teacher	No	44
Ashvini Prasoon	Pune	Doctor	No	43

The frequency matrix on Patient age for the above table can be given as,

Table II. Frequency Matrix

Age	Yes	No
41	1	1
42	1	0
43	0	2
44	0	1

This can be calculated in java with following sudocode

```
ArrayList frequencyage=new ArrayList();
while (for each record){
    if(!frequencyage.contains(age)){
        frequencyage.add(age);
        // update fredata, Yfredata and Nfredata with count as one
    }
    if(frequencyage.contains(age)){
        //get the current count
        // increment count by one
        // update fredata, Yfredata or Nfredata
}
```

Step 2) Generating Submtrices:

In this step, the frequency matrix is divided into the number of submatrix based on the user's input. Each submatrix must have the same number of coordinates calculated based on attribute list input.

For example given the frequency matrix in Table II, if attribute list contains only the "Has Diabetes?" dimension, then the matrix would be split into two submatrices, one of which contains a column age and yes as attributes and other contains age and no.

Step 3) Calculate the wavelet coefficient:

In this step, wavelets are calculated for each submatrix. For calculating wavelets, HAAR wavelet (HW) technique is used. These wavelets are generated by using an array. First all the elements of submatrix are placed into the array. Number of the elements must be equal to power of two. If this condition is not satisfied, we have to add dummy values. This array is copied into another array say copiedarray. The first wavelet coefficient is generated by taking the average of all the values known as base wavelet coefficient. The procedure to calculate remaining HAAR wavelet coefficients of an array of n samples is as follows

(1) Find the average of each pair of samples.(n/2)

(2) Find the difference between each average and the samples it was calculated form.(n/2)

(3) Fill the first half of the array with averages

(4) Fill the second half of the array with differences.

(5) Repeat the process on the first half of the array until n=1.

(6) Finally append the contents of copiedarray array to the newly created array.

Step 4) Calculating error and adding error in wavelet coefficient:

In this step we calculate the error for each wavelet coefficient based on its position in the array and then we add this calculated error in original wavelet coefficient to generate erroneous wavelet coefficients. Amount of error added in wavelet coefficient is given by (error magnitude)/ (weight of wavelet coefficient). Error magnitude is input taken from the user and the weight of wavelet coefficient can be calculated as follows.

For the base coefficient the weight is equal to the number of rows in the frequency matrix. For the other wavelet coefficient weight is 2^{l-i+1}, where l is log(size of the array+1)/ log(2) and i is the position of the node in the array. This position can be calculated for every element of array except contents of appended copiedarray by following sudocode.

```
int p=0;
int y=1;
for(every index)
//calculate minvalue that is tow's power p minus one
//calculate maxvalue that is tow's power q minus one
if(minvalue<=index && maxvalue>=index){
        current_position=y;
//calculate currentcost that is tow's power current_position
        if((currentcost-2)==index){
                p=p+1;
```

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

```
            y=y+1;
        }
}
}
```

Step 5) Reconstruction of submatrices and assembling of submatrices:

This step calculates the submatrices from the erroneous wavelet coefficients and assembles them in order to build the new frequency matrix. This new frequency matrix is nothing but the data to be published. For every index of non-changed appended part of the array, calculation is made in the following way to generate new values.

```
for every index of non-changed appended array
parent_index=index/2
while(parent_index > 0)
if(index%2==0){
        add the value at index to sum
        index=parent_index
        parent_index=parent_index/2
}
else{
        substract the value at index from sum
        index=parent_index
        parent_index=parent_index/2
}
add the sum to base coefficient
```

Due to the above code we will get the published data for single submatrix. This procedure is applied to every submatrix. Combine these submatrices to built final published table.

B. *Privelet+ with heuristic noise reduction*

Due to the sparseness of the natural data, large number of the entries in the frequency matrix may be zero. Also the correlation is exists in the attribute. Due to this correlation adjacent entries in the frequency matrix are close to each other. For example, adjacent entries along the disease dimension of the frequency matrix might not differ much, since people with similar ages are often equally susceptible to the same disease. For these types of the frequency matrices, wavelet coefficients would be small and close to zero. We are adding small error. The resulting value is again a small number. This may suppress the original aim of providing privacy to the data. In order to deal with this problem we have add heuristic noise reduction to privelet+ to create privelet*.

We apply a soft-thresholding techniques based on the θ [Xiaokui Xiao, *et al.* (2011)]. Soft-thresholding transforms each noisy wavelet coefficient c* using a function n_s as follows.

$n_s(c^*,\theta)$ = c* - θ, if c* > θ,
= c* + θ, if c*<θ,
= 0, otherwise.

For calculation of θ we use the following CompThresh algorithm

Algorithm CompThresh (S,λ)

1 . $\sigma^2 = \frac{1}{|S|}\sum_{c*\in S}(c*)^2 - 2.\lambda^2$

2. Sort the noisy coefficient in descending order
3. Store the sorted sequence in an array x
4. For any i ∈ [1,|S|] let $\alpha_i = \sum_{j=1}^{i}(x[i])^2$ and $\beta_i = \sum_{j=1}^{i} x[i]$
5. Perform a linear scan on x to compute α_i and β_i for all the i ∈ [a,|S|]
6. for i=|S| to 1
7. compute the θ that satisfies the following equation
   $\alpha_i - 2\ \beta_i.\ \theta + i.\ \theta 2 = (|S|-1).\sigma 2$
8. if i=|S| and $0 \le \theta \le x[|S|]$ or i<|S| and $x[i] < \theta \le x[i+1]$
   return θ
9. return θ=x[1]

CompThresh first estimates the variance σ^2 of the noise-free coefficients. After that, CompThresh sorts all coefficients in descending order, and stores the sorted sequence in an array X. Next, CompThresh performs a linear scan on X, and computes the following two values for each i ∈ [1; |S|]

1. $\alpha_i = \sum_{j=1}^{i}(x[i])^2$
2. $\beta_i = \sum_{j=1}^{i} x[i]$

Value of the θ can be calculated by following formula,

$\alpha_i - 2\ \beta_i.\ \theta + i.\ \theta^2 = (|S|-1).\sigma^2$

Given X, α_i, and β_i (i ∈ [1; |S|]), CompThresh computes and returns the desired threshold θ and then apply the soft thresholding on noisy wavelets.

IV. EXPERIMENTS

We have implemented a system in java to publish patient database. For this system we have used a patient data set having attributes as Patient Name, Patient Address, Patient Status and Patient Age. This data set includes approximate 10000 records. On this data set we apply Privelet+ method and Privelet* method to calculate answer for the range count query. User enters the error magnitude and selects the attribute list. Then system calculates the published data. This system also measures the average absolute error, root mean squared error and maximum absolute error induced in the system.

The following graph is ranges vs. range count query answer graph. From this graph we can conclude that the error added due to the privelet+ method is less than old method (Dwork's method) for range count query answer. Privelet* method provide more accurate results than both these method.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 10, October 2013

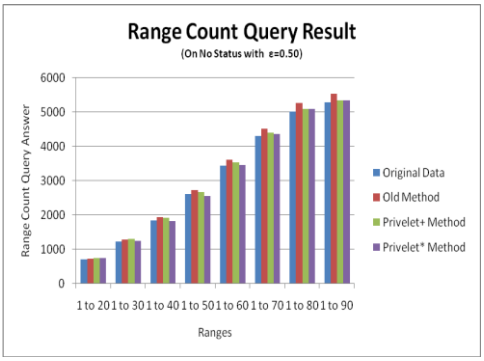

Figure 1 Range Count Query Result

The quality of each approximate answer of range count query is gauged by its absolute error and relative error with respect to the actual query result. The absolute error of X is defined as |X-act| and relative error of X is computed as |X-act|/max{act,s}; where s is sanity bound that mitigates the effects of queries with excessively small selectivities. We set s as 0.1 percent of the number of tuples in the data set. We measure the absolute error and relative error across the coverage and selectivity of the query respectively. Selectivity of query q is the fraction of tuples in the data set that satisfy all predicated in q and coverage of q is the fraction of entries in the frequency matrix that are covered by q.

The following graph shows the average absolute error incurred in answering each query. The X-axes of the graph represents the coverage of the query. The graph is for PStatus as no and the ε as 0.50.

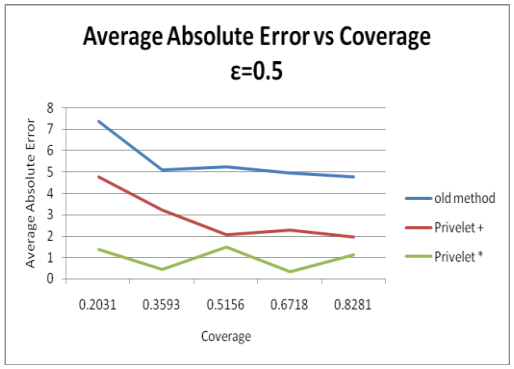

Figure 2 Average Absolute Error vs. Coverage

The following figure shows the average relative error incurred in answering each query. The X-axes of the graph represents the selectivity of the query. The graph is for PStatus as no and the ε as 0.50.

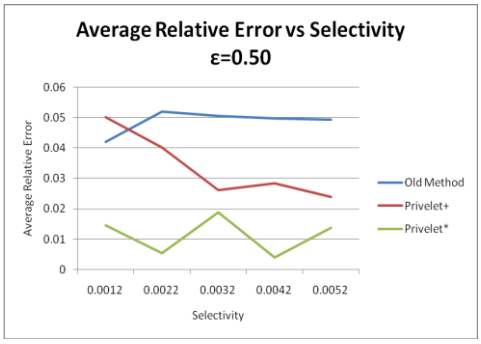

Figure 3 Average Relative Error vs. Selectivity

From the above figures we can conclude that amount of error added in privelet* method is less than privelet+ and old method. Privelet* provides more accurate results than other two methods.

V. CONCLUSION

Privelet+ and Privelet* methods are data publishing techniques. Privelet+ and Privelet* methods provide improved results over a range count query applied. Experimental results provide the effectiveness of the Privelet+ method and privelet* method.

REFERENCE

[1] Carlisle D. M., et. al. (2007): California inpatient data reporting manual, medical information reporting for California (5th Ed) Tech. rep., Office of Statewide Health Planning and Development.

[2] Dwork C., et. al. (2006): Calibrating Noise to Sensitivity in Private Data Analysis, Proc. Third Theory of Cryptography Conf. (TCC), pp. 265-284.

[3] Dwork Cynthia (2008): Differential Privacy: A Survey of Results, TAMC 2008, LNCS 4978, pp. 1–19,

[4] Fung B.C.M., et al, (2010): Privacy-Preserving Data Publishing: A Survey of Recent Developments, ACM Computing Surveys, vol. 42,no. 4, pp. 14:1-53

[5] Garofalkis M., Amit Kumar (2005): Wavelet Synopses for General Error Metrics, ACM Transactions on Database Systems, Vol. 30, No. 4, Pages 888–928.

[6] McSherry F. and Mironov I.(2009): Differentially private recommender systems: Building privacy into the netflix prize contenders., In KDD.

[7] Rastogi V. and Nath. S., (2010): Differentially private aggregation of distributed time-series with transformation and encryption, In SIGMOD.

[8] Xiaokui Xiao, Guozhang Wang, and Johannes Gehrke(2011): Differential Privacy via Wavelet Transforms, IEEE Transactions on knowledge and data engineering, Vol. 23, No. 8.