CS297 Report

Adding Differential Privacy to an Open Source Discussion Board System

By

Pragya Rana

pragya.rana9@gmail.com

Project Advisor: Dr. Chris Pollett

Department of Computer Science

San José State University

One Washington Square San José, CA 95112

TABLE OF CONTENTS

INTRODUCTION **3**
DELIVERABLE 1 ..... **4**
DELIVERABLE 2 ..... **6**
DELIVERABLE 3 ..... **8**
DELIVERABLE 4 ..... **10**
CONCLUSION **12**
REFERENCES **13**

INTRODUCTION

This project deals with implementing a privacy system for statistics generated by Yioop search engine and discussion board system. Currently in Yioop's discussion board system, various statistical data are calculated such as number of users belonging to a group, number of views of a thread, etc. Statistical data provides information based on a collective data or a sample. When statistical data is made publicly available, there is no guarantee of preserving the privacy of an individual.

For example, given the statistics of annual income of individuals of a certain place and average annual income. If one person knows that someone's annual income is higher than the average income by certain percentage, he/she can easily calculate this person's annual income. Any data extracted should not reveal any sensitive information of an individual.

Therefore, the motivation of this project is to protect the privacy of an individual who is part of the statistical database of Yioop's search engine and discussion board system. The goal of privacy-preserving statistical database is to allow the user to learn properties of the extracted information, while protecting the privacy of the individual in the sample. In order to achieve this, there is a privacy mechanism called Differential Privacy that we will implement in this project.

Differential privacy preserves the privacy up to some controllable parameters of individuals when statistics from a database are made public. With this measure, accurate information about the database is provided while at the same time, privacy of the individual is maintained.

There are few ways of achieving differential privacy of data. One of the mechanisms is called **ε-differential** privacy (Dwork, 2006). This privacy is achieved by adding some appropriately chosen random noise to the query's answer in such a way that the information retrieved by the user is still accurate and at the same time no sensitive information is leaked about an individual.

The rest of the report explains about the work done in each of the four deliverables for CS297. In Deliverable 1, I prepared a presentation on Differential Privacy based on Cynthia Dwork's paper. In Deliverable 2, I added a statistical chart

when clicked on most visited threads in Yioop. In Deliverable 3, I developed a test suite of statistical attacks against query and discussion board statistics. And finally in Deliverable 4, I added differential privacy to number of views of each group's thread.

DELIVERABLE 1

The first deliverable was to understand the concept of Differential Privacy, the need for this mechanism in Yioop search engine and then prepare a presentation on the topic. Before implementing differential privacy in Yioop, there are few things that need to be understood such as the model used in computation of the privacy, basic techniques and theorems used to achieve differential privacy.

Differential privacy ensures users participating in any form of data analysis by protecting their sensitive information. This doesn't mean the data will be less accurate by hiding information about the user's data. We will still be able to reveal useful information from the statistical database, while protecting the privacy of the individuals in the sample.

Privacy Data Analysis

There are two models for privacy mechanisms. In Non-Interactive Setting data collector (a trusted entity) publishes a "sanitized" version of the collected data (sanitization employs techniques such as data perturbation and sub-sampling, removing well-known identifiers such as names, birthdates, ssn). And in Interactive Setting data collector provides an interface through which users may pose queries about the data and get answers.

While comparing Non- Interactive with Interactive approach, the results for Interactive approach are powerful. Non- Interactive approach is more difficult due to difficulty of supplying utility that has not yet been specified at the time the sanitization is carried out.

Differential Privacy

A randomized function K gives ε-differential privacy if for all data sets D1 and D2 differing on at most one element, and all $S \subseteq \text{Range}(K)$,

$\Pr[K(D_1) \in S] \leq \exp(\varepsilon) \times \Pr[K(D_2) \in S]$

A mechanism K satisfying this definition ensures the user participating in the database that any responses to queries is equally likely to occur even if the participant removed his/her data from the data set. For example, if an insurance provider refers to a database before making a decision of giving an insurance to certain user x, then the impact on the user x will remain the same whether or not user x opts in or out of the database.

Achieving Differential Privacy

This section describes a concrete interactive privacy mechanism for achieving ε-differential privacy. The mechanism works by adding appropriately chosen random noise to the answer a = f(X), where f is the query function and X is the database.

Exponential Noise and the L1-Sensitivity

In order to achieve ε-differential privacy, there is an addition of random noise whose magnitude is chosen as a function of the largest change a single participant could have on the output to the query function; we refer to this quantity as the sensitivity of the function. Sensitivity is a property of the function alone that is independent of the database. The technique is best when Δf is small, i.e. introduce the least noise.

For $f : D \rightarrow R^d$, the L1-sensitivity of f is

$$\Delta f = \max_{D1,\, D2} \| f(D1) - f(D2) \|$$

for all D1,D2 differing in at most one element.

The privacy mechanism, denoted as K_f for a query function f, computes f(X) and adds noise with a scaled symmetric exponential distribution with variance σ^2 in each component, described by the density function

$$\Pr[K_f(X) = a] \propto \exp(- \| f(X) - a \| 1/\sigma)$$

For $f : D \rightarrow R^d$, the mechanism K_f gives $(\Delta f/\sigma)$- differential privacy.

For query strategy $F = \{f\rho : D \rightarrow R^d\}$, the mechanism K_f gives $(\Delta F/\sigma)$-differential privacy.

Deliverable 2

The second deliverable was to add statistical chart when clicked on most visited threads in Yioop. This deliverable helped in getting familiar with Yioop coding framework and thus worked as learning curve for the rest of the deliverables. Currently, the statistics of the group including its threads and wikis are shown in terms of total number of views. One can see total views for each group, thread and wikis during the last hour, day, month and year.

This deliverable extends the feature by showing the statistics in a chart. The chart spans total views into different time periods such as during the last 24 hours in a day, 30 days in a month or 12 months in a year. There is a new link added to number of views in Group Statistics page. This links to a new page for viewing the chart showing the statistics of each group item. The link is added for ‘last day’, ‘last month’ and ‘last year’. When you click on ‘last day’, it shows the chart of number of views during the last 24 hours. Similarly, when you click on ‘last month’, it shows the chart of number of views during the last 31 days. And when you click on ‘last year’, it shows the chart of number of views during the last 12 months of the year. Since, ‘last hour’ and ‘All time’ views are just one data, there is no link for these two statistics.

There are few changes made in ImpressionModel. The current SQL statements that calculate total views for each time period have been slightly modified. Also a new function called ‘getPeriodHistogramData is added in ImpressionModel that calculates total number of views of given item for given time period. This is called from

SocialComponent, which in turn uses the returned data from ImpressionModel, formats it and prepares the data that is then displayed via ManagegroupsElement.

In order to view the latest statistics, make sure to turn on the Media Updater, which is only accessible by root. Go to Manage Machines -> Media Updater and click on 'Log On' to turn it on. Once the Media Updater is turned on, you can view the statistical chart via Manage Account -> Click on any group -> Click Statistics. This can also be viewed through Manage Groups page.

The following chart displays statistics of group 'TestGroup1' about the number of views in the last day.

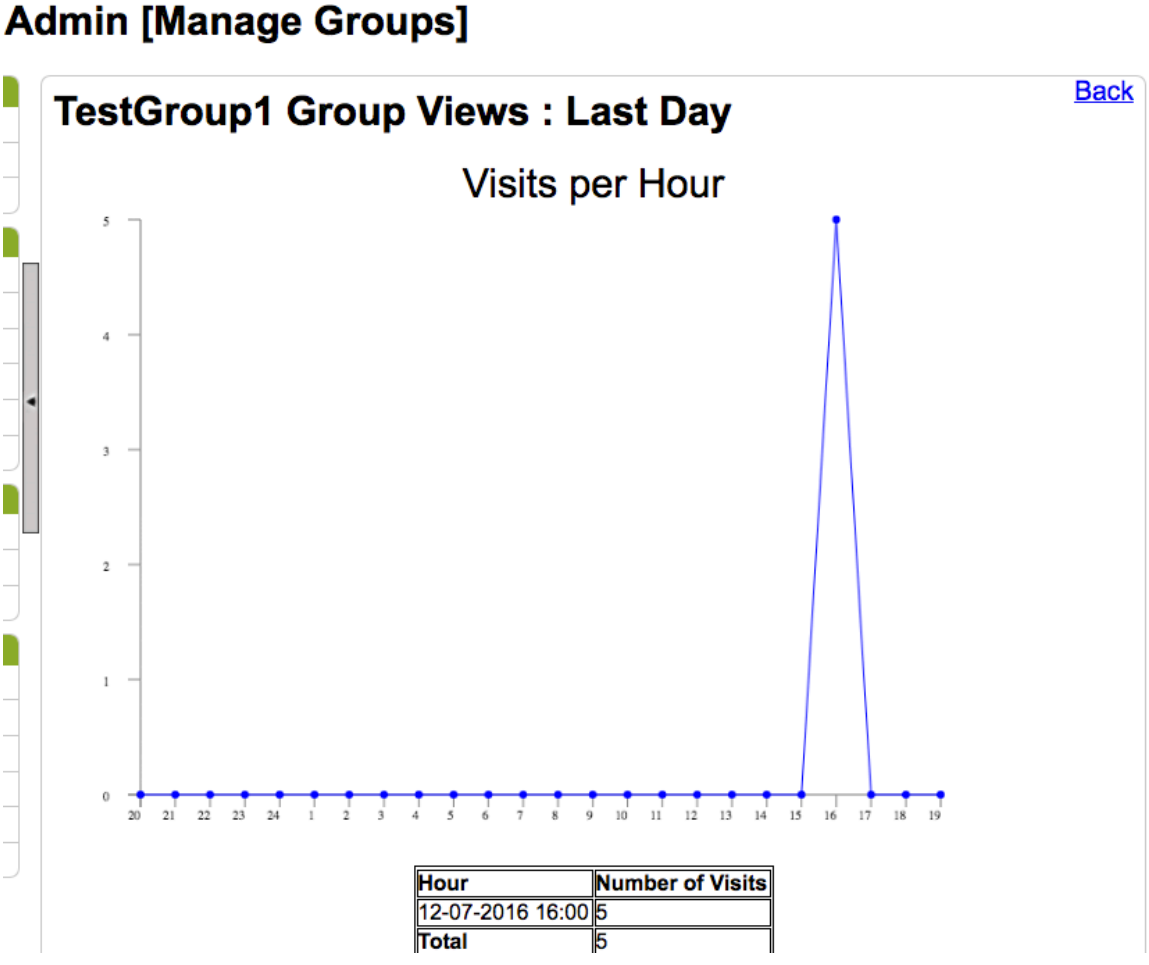

Figure 1: Statistical Chart

Deliverable 3

The third deliverable was to develop a test suite of statistical attacks against query and discussion board statistics. We needed to come up with some test suites so that we could identify the areas in Yioop where we could apply differential privacy for this project.

Statistical attack is a method of deriving sensitive data from non-sensitive data. The number of entries returned by any query in a statistical database is either zero or any number up to total number of records stored on the database. The expected behavior of statistical query is to return non-individual results. Even though the results returned by the query don't reveal data about a single individual or a record, it is possible to infer protected information by combining the results of several statistical queries into one. These attacks depend on the specific aggregate operator they use.

There are two cases of inference, which often appear in databases: **data aggregation and data association (**Burtescu, 2009). **Data aggregation** problem arises whenever a set of information is classified at a higher level than individual levels of involved data. **Data association** problem arises whenever two values taken together are classified at a higher level than the one of each value.

Types of attacks:

1. **Direct attacks** are attacks that involve queries, which directly yields sensitive data item. These are successful only if the database does not implement any protection mechanism.
2. **Indirect attacks** derive sensitive data from non-sensitive statistical results. These are attacks that are executed by combining multiple queries to extract other data than those that are displayed by individual query.

Test suite of statistical attacks against query and discussion board statistics

1. Finding out the user who has viewed a certain thread: When a user belongs to a certain group and there are only two members in that group, then one user can easily figure out whether or not the other user has recently viewed any thread belonging to that certain group. The total number of views of any thread is always visible. This can be accessed via Group Feed page. Let's say the current view of a thread is 100 and after refreshing it, it goes up to 101, then it's obvious that another user has just viewed that thread.

2. Finding out the user who has posted in a group's thread: Let's say any user can post to any thread of a group as an anonymous. But if there are only 2 users belonging to a certain group, even though a user posts as an anonymous, this user is revealed to another user.

3. We can find out user id of a user who has posted any comments on any group discussion board. For example:
    - Click on any group (say 'TestGroup1'). This shows list of current posts under that group.
    - Below each post, you can see 'Last Post: <DD/MM/YYYY> - <USER_ID>'.
    - For the post (say 'Welcome to TestGroup1 Thread'), you can see 'Last Post: 17/09/2016 - testuser1'. Click on 'testuser1'. You can get the user_id in the url or under 'Query Statistics':

    SELECT COUNT(DISTINCT GI.ID) AS NUM FROM GROUP_ITEM GI, GROUPS G, USER_GROUP UG, USERS P WHERE P.USER_ID='3' AND ….

4. Indirect attack via SQL Query

If we know that one user (user_id 2) has only 1 advertisement, we can find budget put by user_id 2

a. SELECT SUM(BUDGET) FROM Advertisement WHERE user_id != 2;
b. SELECT SUM(BUDGET) FROM Advertisement

By taking a difference of (a) and (b), we can get the Budget put by user_id 2

Deliverable 4

Deliverable 4 starts the first implementation of Differential privacy. It adds differential privacy to number of views of each group's thread. Basically, it fuzzifies the number of views of each thread of the group by using epsilon differential privacy according to Cynthia Dwork's paper on Differential Privacy. This is done by first checking the current code that calculates the number of views of each thread. Then some noise is added to the current implementation.

A new function addDifferentialPrivacy is added to the base controller that actually implements the ε-differential privacy. The privacy mechanism, denoted as K_f for a query function f, computes f(X) and adds noise with a scaled symmetric exponential distribution with variance σ^2 in each component, described by the density function

$$Pr[K_f(X) = a] = \exp(- || f(X) - a || 1/\sigma)$$

Here, if the actual value is n, the value of 'a' is taken between the range 0 and 2n + 1. Then an integrated value is calculated within that range for the equation above. And a random value is selected between 0 and the integrated value in order to get fuzzified result.

An experiment is conducted based on the equation above to check how much the calculated view deviates from actual number of views. This experiment uses different x-axis values for the number of views starting with 1, 5, 25, 625, 3125, 15625, 78125,

390625 and 1953125. For each x-axis value, there are 50 y-axis values that are generated by the equation above. The following chart uses the scatter plot that clearly shows how y-axis values are scattered for each x-axis value. As the value of x-axis gets larger, y-axis values come closer.

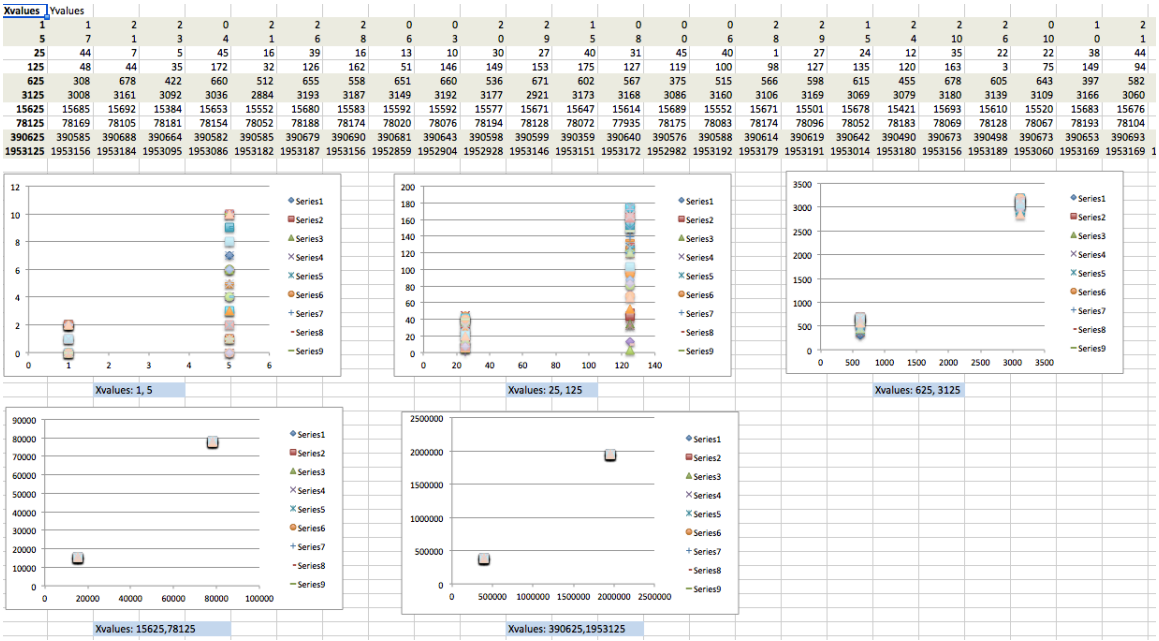

Figure 2: Fuzzy Data Experiment

The function addDifferentialPrivacy is called from controller's SocialComponent. There is also an update in Item_Impression_Summary table. Now there are two additional columns FUZZY_NUM_VIEWS and TMP_NUM_VIEWS. A flag called 'DIFFERENTIAL_PRIVACY' is also added in Config.php to turn on/off differential privacy as it may be required to use actual data instead of fuzzified data.

CONCLUSION

In conclusion, it was a good experience working on an interesting topic of 'Differential Privacy'. I get to learn about the concept of differential privacy and the importance of privacy-preserving statistical database. Especially when today's technological world requires so much of statistical data in every field, it becomes extremely important to not reveal any individual's sensitive information while still be able to continue working accurately with the statistical data.

While working on this project, I got familiar with Yioop's search engine and the coding framework used in this project. I learned how current statistics about the group is computed, stored in the database and then fetched from the database. I used this knowledge to develop a chart that displays the statistics of the group such as number of views of groups, threads and wikis during the last hour, week, month and year. Then I came up with some use cases of statistical attacks against query and discussion board statistics. This gave an idea of where in Yioop can we possibly apply differential privacy. The first place where I started an actual implementation of differential privacy was in Group feeds page where the total number of views of each thread is displayed. Now this piece of information has been fuzzified in order to prevent any one to figure out actual user's viewing activity.

For the next semester, we will explore in depth about other model, techniques and theorems used in the computation of the privacy in order to achieve differential privacy. We can also identify any possible areas in Yioop search engine where we can implement differential privacy besides the number of views that was implemented this semester. We need to do further research on different mechanisms mentioned by Cynthia Dwork in her paper and use an appropriate mechanism in Yioop.

REFERENCES

Dwork, C. Differential Privacy, 33rd International Colloquium on Automata, Languages and Programming, part II, 2006

Dwork, C. and Roth, A. The Algorithmic Foundations of Differential Privacy, Foundations and Trends in Theoretical Computer Science Vol. 9, Nos. 3–4 (2014) 211–407, 2014

Burtescu, (2009). Database Security – Attacks and Control Methods. Retrieved from http://www.jaqm.ro/issues/volume-4,issue-4/pdfs/burtescu.pdf