Spring 5-26-2017

Adding Differential Privacy in an Open Board Discussion Board System

Pragya Rana
San Jose State University

Adding Differential Privacy in an Open Board Discussion Board System

A Project Presented to

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment of

the Requirements of the Degree

Master of Science

By

Pragya Rana

May 2017

The Designated Project Committee Approves the Master's Project Titled

Adding Differential Privacy in an Open Board Discussion Board System

By

Pragya Rana

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSE STATE UNIVERSITY

May 2017

Dr. Chris Pollett, Department of Computer Science Date

Dr. Melody Moh, Department of Computer Science Date

Mr. Mahesh Subedi, Clari Inc. Date

APPROVED FOR THE UNIVERSITY

Associate Dean Office of Graduate Studies and Research Date

ABSTRACT

Adding Differential Privacy in an Open Discussion Board System

This project implements a privacy system for statistics generated by the Yioop search and discussion board system. Statistical data for such a system consists of various counts, sums, and averages that might be displayed for groups, threads, etc. When statistical data is made publicly available, there is no guarantee of preserving the privacy of an individual. Ideally, any data extracted should not reveal any sensitive information about an individual. In order to help achieve this, we implemented a Differential Privacy mechanism for Yioop. Differential privacy preserves privacy up to some controllable parameters of the number of items or individuals being aggregated when statistics from a database are made public. With this measure, reasonably accurate information about the database is provided while at the same time, privacy of the individual is maintained. The privacy mechanism called ε-differential privacy (Dwork, 2006) achieves this 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. We implemented Differential Privacy for group's statistics page. These pages display provide various statistics about a discussion group including its threads and wikis. We also implemented ε-differential privacy for query statistics page that displays the statistics about each query entered by a user in the Yioop's search bar. The project also adds an additional level of privacy by using encryption of an application level database information to secure some sensitive data.

ACKNOWLEDGEMENTS

I would first like to thank my project advisor Dr. Chris Pollett of the Department of Computer Science at San Jose State University. Dr. Pollett consistently guided me in the right direction whenever I had some doubts or got stuck in my project.

I would also like to thank the committee members, Dr. Melody Moh and Mr. Mahesh Subedi, for their time and input without which this project could not have been successfully completed.

Finally, I would like to express my gratitude to my parents, friends and family for providing me with love, support and encouragement.

TABLE OF FIGURES

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Today there are various online platforms provided to users for various purposes such as social media, online shopping, online movies, etc. These platforms use users' personal data to generate statistics. By collecting user data online, they can identify different types of users both to improve the site itself and also to target users to sell them goods and services. For example, Amazon recommends the products for buying based on the customers browsing history. Therefore, protecting user's private information is critical in today's online world.

This project implements privacy techniques on a statistical database. Users whose data appears in such a database should not be subject to any uniquely identifying data even if the statistics are made public. There have been many approaches for privacy when organizations publish personal statistics such as removing very personal information about the individual (dob, ssn) and making the user as anonymous as possible. But even with this approach, there had been cases in the past when releasing anonymized data had failed to preserve the privacy of an individual. For example, there was the identification of medical records of the governor of Massachusetts in a public "anonymized" medical database. There was also the identification of search history of Thelma Arnold in public "anonymized" AOL query records [8].

There are numerous cases of attacks happening on the database systems on a frequent basis. Relying on older ways of authentication and access control are not enough to protect the data from malicious users. Therefore, there is a critical need to protect the privacy of an individual who is part of the statistical database.

It is possible that even the user who is part of the system may have bad intention and can try to derive some sensitive data from the non-sensitive statistical results displayed by the system. Thus, we want a level of privacy in a statistical database that makes it difficult for anyone to extract any information from the statistics.

One fundamental technique in order to achieve such privacy is called ε-differential privacy (Dwork, 2006). It is the privacy mechanism that is implemented in this project. 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.

In order to add differential privacy, we first looked at different components of the Yioop open source discussion system and then identified the most critical components that contain more sensitive data and is therefore vulnerable to statistical attacks. Then we applied the concept of differential privacy to those components. We also looked at other techniques to improve the privacy in the database system of Yioop. One of the techniques we implemented was database encryption at an application level.

This report is divided into seven chapters: Chapter 2 gives the background about differential privacy and also discusses the previous research done to achieve differential privacy in database systems. It also mentions other techniques used to improve the privacy in Yioop discussion board system. Chapter 3 discusses the preliminary work for this project that was done such as database changes and other functionalities. Chapter 4 gives an overview about the design of our differential privacy system as well as how we designed our encryption scheme for our the database system. Chapter 5 gives an overview of the implementation of the functionalities discussed in the design section. Chapter 6 discusses different types of testing and experiments conducted to test the new changes in different components. It also describes some tests we ran to check the performance of the system after adding the new changes. Finally, Chapter 7 has a conclusion.

CHAPTER 2
BACKGROUND

This project considers techniques to improve privacy in the Yioop discussion board system. One of those techniques is Differential Privacy. Another is database encryption.

Dwork (2006) defines Differential Privacy as a randomized function K that gives ε-differential privacy if for all data sets D_1 and D_2 differing on at most one element, and all $S \subseteq$ Range(K),

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

A mechanism K that satisfies above definition ensures the user participating in the database that any responses to queries is equally likely to occur even if the participant decides to remove his/her data from the data set (Dwork, 2006, p. 9). 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 (Dwork, 2006, p. 9).

There are two models for privacy mechanisms. In Non-Interactive Setting the data collector who is a trusted entity publishes a sanitized version of the collected data by removing well-known identifiers such as names, data of birth, ssn) (Dwork, 2006, p.3). And in Interactive Setting the data collector provides an interface through which users may present queries about the data to get answers with some added noise. (Dwork, 2006, p.3).

An interactive privacy mechanism is used 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.” (Dwork, 2006, p.9)

In addition to the algorithms discussed above, Database Encryption is also used to protect the data in a database from intruders. Database Encryption uses an algorithm to cipher the real data in a database. These transformed data needs to be decrypted first in order to use them. There are different techniques available for database encryption few of which are mentioned below (Database Encryption):

- External Database Encryption
- Column Level Encryption
- Field-Level Encryption
- Symmetric/Asymmetric Database Encryption
- Application Level Encryption

For this project we have used application level encryption where we are using symmetric keys which are stored in an external database. We are using column level and field level encryption. This way we can control which data is more sensitive and perform encryption only in those data. This is different with encryption used in networks or storage systems where data is fully encrypted or not encrypted at all.

There have been different types of work done in the database level in order to secure the database. One of the researches presented in the paper by Patel, Sharma and Eirinaki is called Negative Database for Database Security. A Negative Database contains data that includes

real data as well as negative data. The framework proposed in the paper manipulates the actual data and stores in the database which can then be retrieved in an efficient way. Thus, it makes it harder for malicious users to extract real information even though they may be able to get access to the database. The security framework consists of four main modules, Database caching, Virtual database encryption, Database encryption algorithm, and Negative Database conversion algorithm (Patel et. al). After passing through these modules, the manipulated data is then stored in the database. The following is an overview of each module described by Patel et. al.

- Database Caching: The frequently used data is stored in an easy access memory such as RAM thus making it faster to access data and increases the performance.

- Virtual Database Encryption: A set of randomly generated keys are added to the data in the form of objects and attributes which are then sent to the database.

- Database Encryption Algorithm: The public key encryption algorithm RSA is used. The encryption algorithm is followed by Database caching and Virtual Database Encryption algorithm. This module uses the input from prior modules and encrypts the data before passing it to the next layer, Negative Database Conversion layer.

- Negative Database Conversion Algorithm: The is the last module which creates a large set of values rather than just a single tuple. The generated sets of data are inserted in the database.

For this project, we have used some concept of negative database by adding some random noise to the actual data and encrypting them before storing in the database. We will discuss about this in detail in the implementation section.

CHAPTER 3

PRELIMINARY WORK

This section describes about the preliminary work that were done in order to implement Differential Privacy for this project. Before getting into differential privacy, there were few things that needed to be understood such as the model used in computation of the privacy, basic techniques and theorems used to achieve differential privacy.

In the paper presented by Dwork (2006), we chose ε- differential privacy for achieving the differential privacy. According to her, 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

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

In order to achieve ε-Differential Privacy, $\sigma \geq \epsilon / \delta f$ where Δf is the L1-Sensitivity of a function.

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.

Once we knew which algorithm to use, we extended existing feature in Yioop about the statistics of the Discussion Board System by adding a chart that displays statistics in more depth.

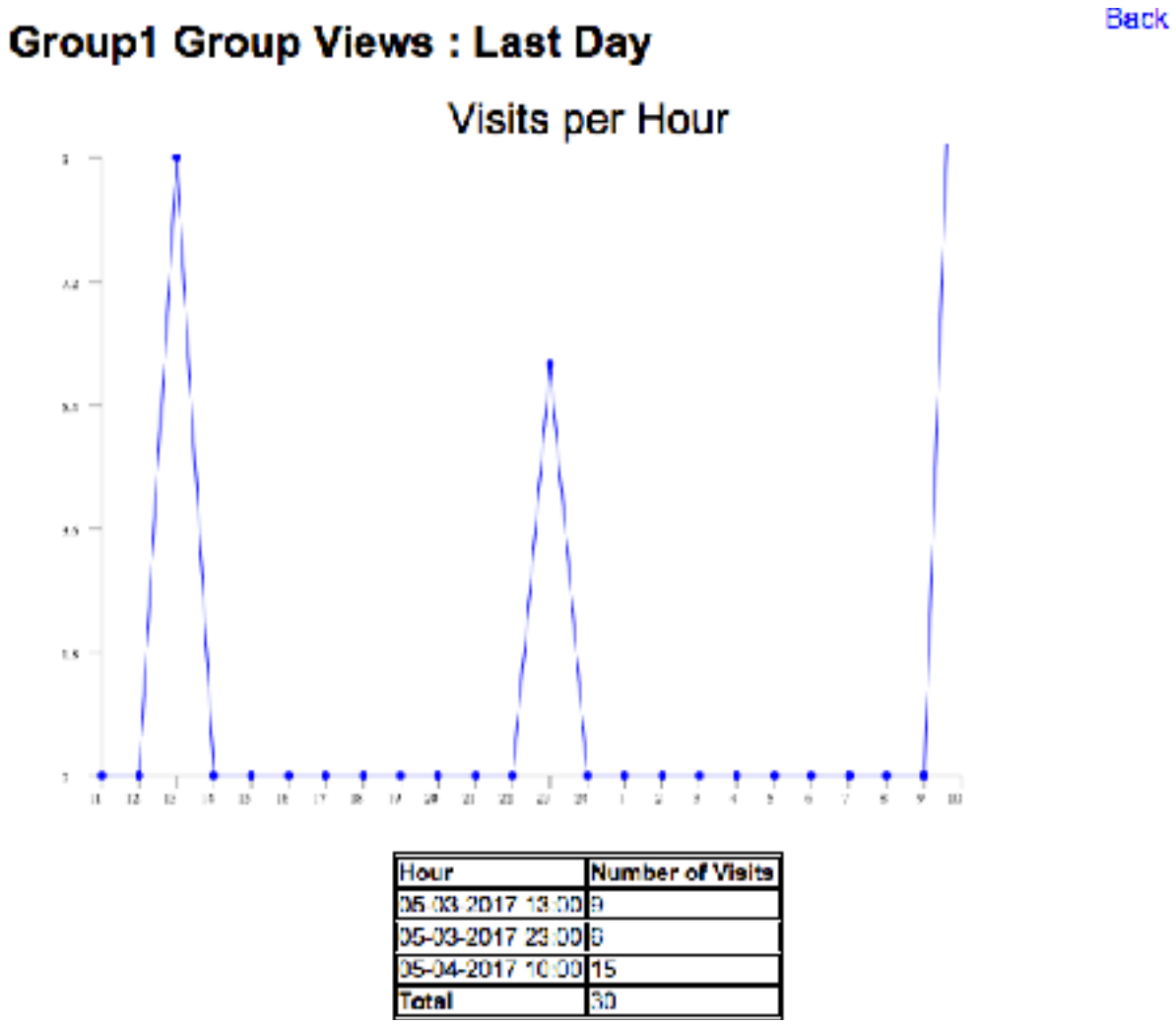

Figure 1: Statistical chart showing number of visits during last 24 hours

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 30 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.

We then needed to identify areas in Yioop search engine as well Open Source Discussion Board where we could apply Differential Privacy. So we developed a test suite of statistical attacks against query and discussion board statistics. Statistical attack is a method of deriving sensitive data from non-sensitive data (Burtescu, 2009).

An example of a test suite of statistical attack is 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.

The first implementation of the algorithm was done by adding 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 ε-differential privacy according to the paper by Dwork (2006). 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. There were also changes made in the database where the analytics compute the statistics and stores the result in the database. We updated some tables to store values that were calculated by using differential privacy algorithm.

CHAPTER 4

DESIGN

This section discusses the design part of Adding Differential Privacy to an Open Source Discussion Board System.

Defining policy about specific data that is targeted for adding privacy

Yioop's search engine and Discussion Board System contains different types of contents under groups, threads, wikis, search tool. Each group contains threads and wikis. There will be number of users who can join any group with or without the owner's approval depending on the settings of the group. Each user can create a new thread, comment on a thread, add/edit/delete a thread depending on the user's access level.

There are different analytics that are run to compute different types statistics. For example, we can view the statistics of each group. One can see total views for each group, and all its threads and wikis during the last hour, day, month and year. Also, there is a search analytics that computes statistics about each query entered by a user in the search.

There are different types of data that gives various information about each component. Thus we need to define policy which identifies specific data that requires higher level of privacy. After identifying special cases of data, we can focus on those sets of data to add more privacy than before.

Controlling Security Feature from the UI level

After the preliminary work was done, the only way to enable Differential Privacy was through the code. In config file, you could change the value of Differential Privacy to either 1 which will enable or 0 which will disable the setting. There was no way of specifying it via the user interface. There should be an option for the root user to enable or disable Differential Privacy via the user interface. So, an enhancement in the security feature in the user interface is needed. An option should be added for Differential Privacy under Security section of the Yioop search engine where the root user could select either ‘Enable’ or ‘Disable’ option. Differential Privacy should then be added to the data that is part of the privacy policy as mentioned in the previous section based on the option selected by the user.

Database encryption at an application level

To increase the level of privacy, some of the data can be secured even more by encrypting them in the database. This also requires defining policy such as which data is more sensitive and requires higher security and perform encryption only in those data. Once the set of data has been identified, we need to select the type of encryption to use for encrypting and decrypting the data. For this project we are using application level encryption where symmetric keys are stored in an external database. We are using column level and field level encryption. We are using an external database so that even if an intruder gets access to the main database, he/she will not be able to decrypt the data in the database without having access to the external database. Thus this adds additional level of security to the database.

Adding Differential Privacy according to the defined policy on selected data

The analytics job uses raw data accumulated for each group based on activities of all the users. It then aggregates those data into different time periods giving the statistics hourly, daily, monthly and yearly. These are the impression statistics. This statistics can give information on how frequently certain group or thread or wiki is visited.

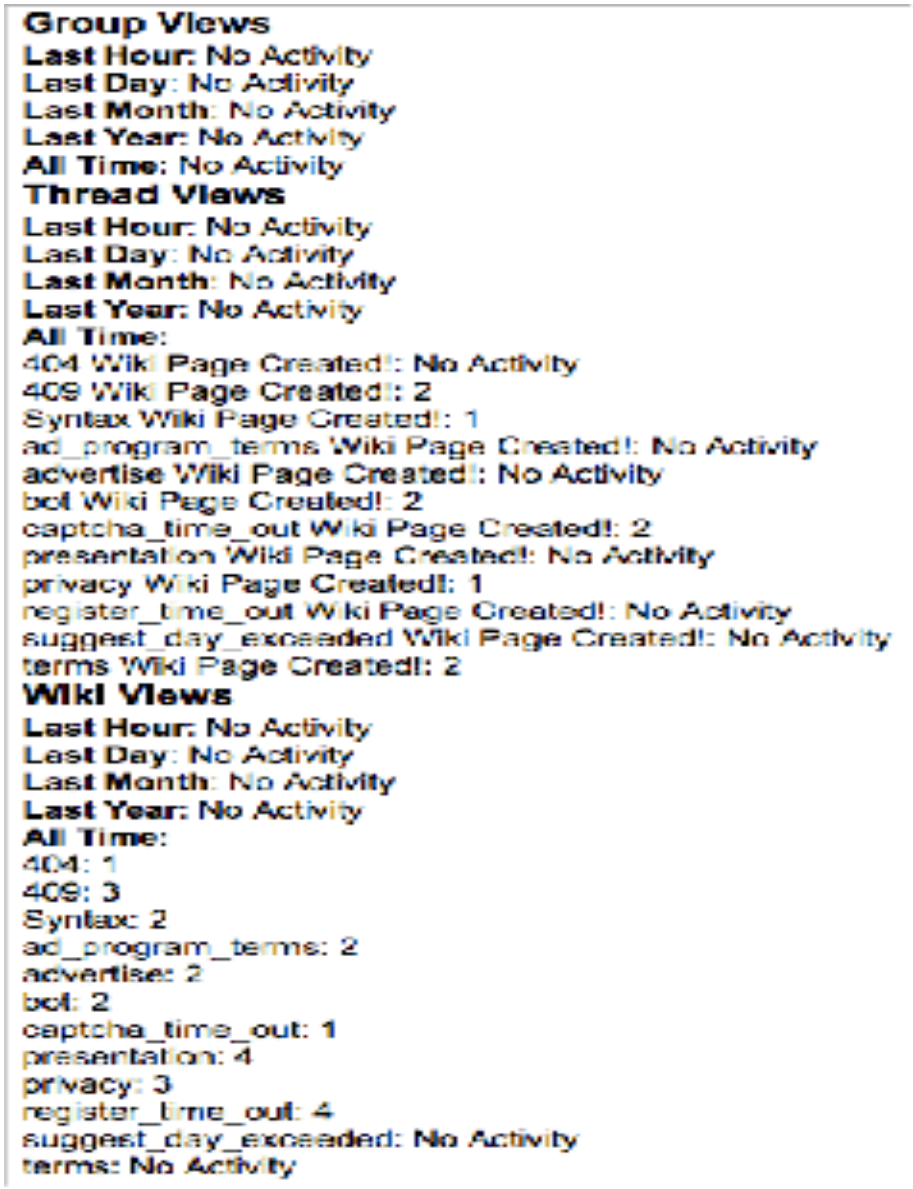

Figure 2: Groups Statistics Summary Page

In addition to the main summary page as shown above in the figure, one can also view the statistics in the chart for each time period. This displays statistical information in more detail. For non-sensitive group, it is reasonable to display the data to users. But when it comes to sensitive data, it becomes critical to protect the data of an individual.

CHAPTER 5

IMPLEMENTATION

Controlling Security from the UI level

The root user can control the level of privacy via the user interface. An option to enable or disable the differential privacy has been added under Security section. The user can navigate to Security section from the left hand side menu. Then under Privacy section, a dropdown option has been added for Differential Privacy. The drop down option includes Enable and Disable. By default Disabled option is selected. Selecting the option 'Enable' will enable the differential privacy in the Yioop search engine and Discussion Board System. Similarly, selecting the option 'Disable' will disable the differential privacy in the Yioop search engine and Discussion Board System.

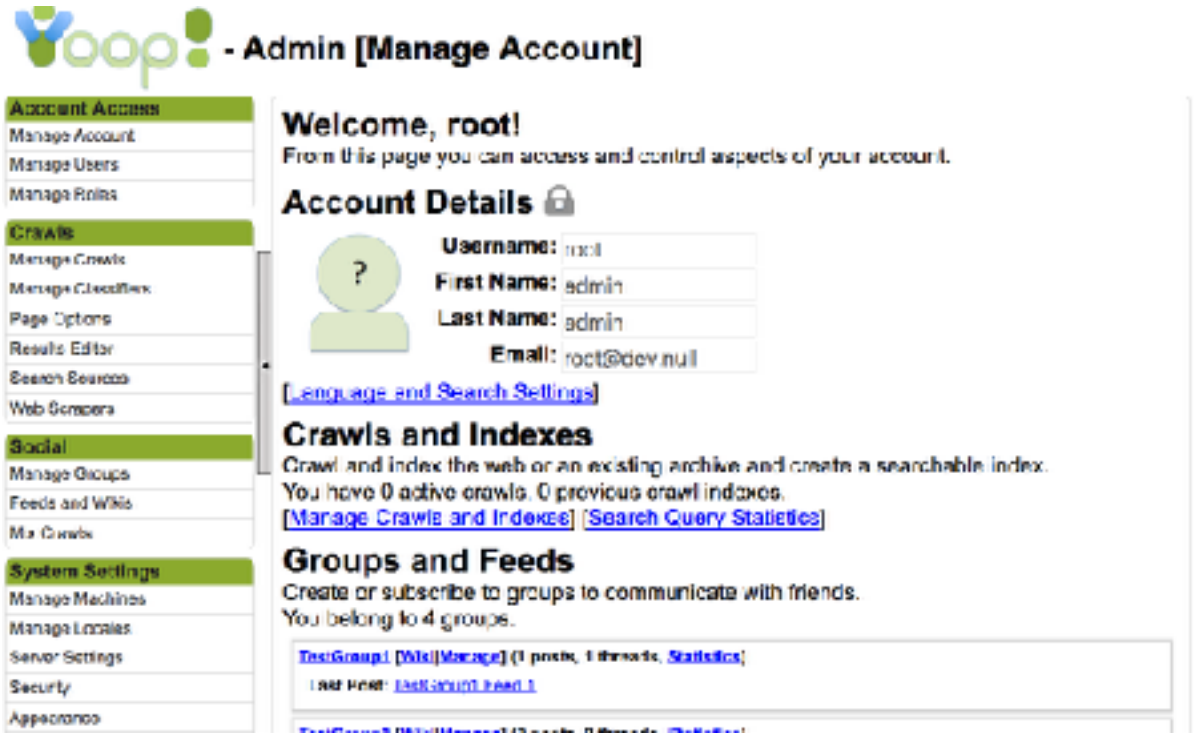

Figure 3: Manage Account

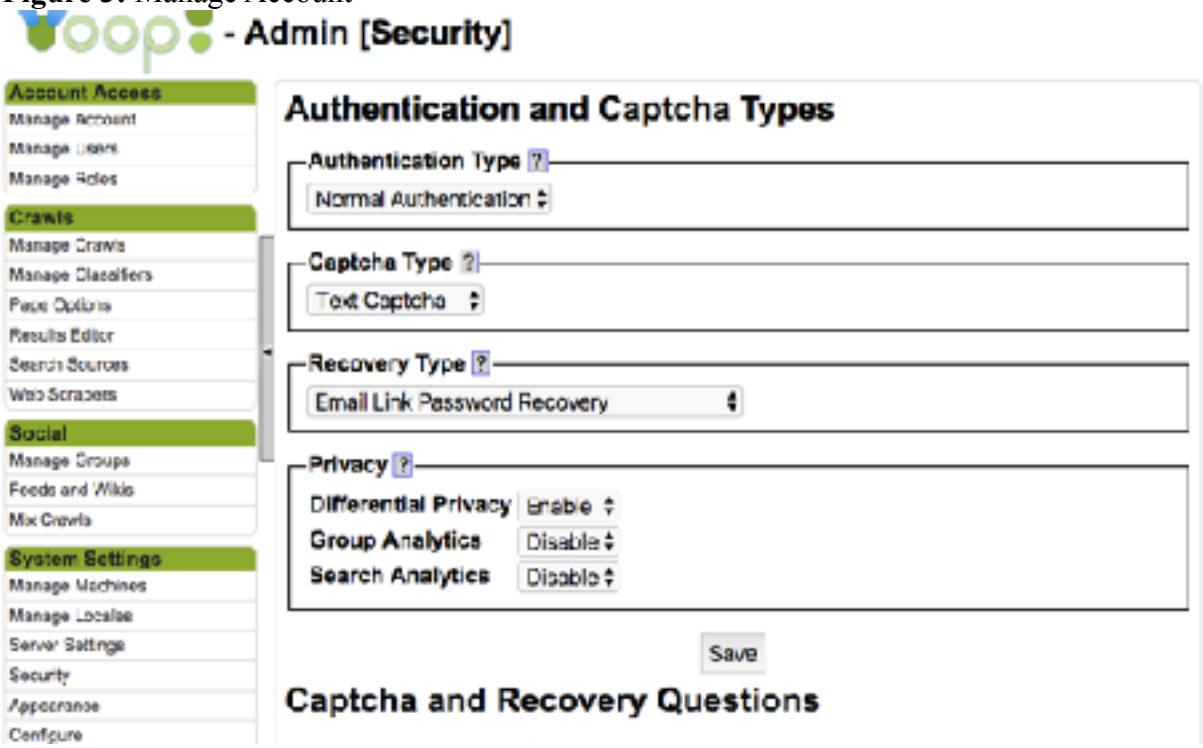

Figure 4: Security Settings

Database encryption at the application level

We identified sets of data in the open source discussion board system that require more control in terms of privacy. In the discussion board, there are different types of groups. Each group has a list of users and users can post different threads, add/edit/delete comments, vote plus or minus for each thread, etc. Thus we are adding encryption to certain data inside each group. There are different types of encryption that can be performed in the database. We are using application level encryption of the database. Not all data inside each group needs to be encrypted as some of them are very insensitive. We are interested in encrypting all the threads posted by all the users and it's replies/comments. Hence, we are using column level and field level encryption.

In order to perform encryption and decryption we are using symmetric keys. The keys are stored in an external database. By doing so, the data cannot be decrypted without having access to the keys which are stored in an external database. So even if the intruder gets access to the keys which are stored in an external database. So even if the intruder gets access to the main database system, he/she will not be able to get the plain data. To make it more secure, we are injecting random values to the actual data every time we are performing any database operation (insert, update).

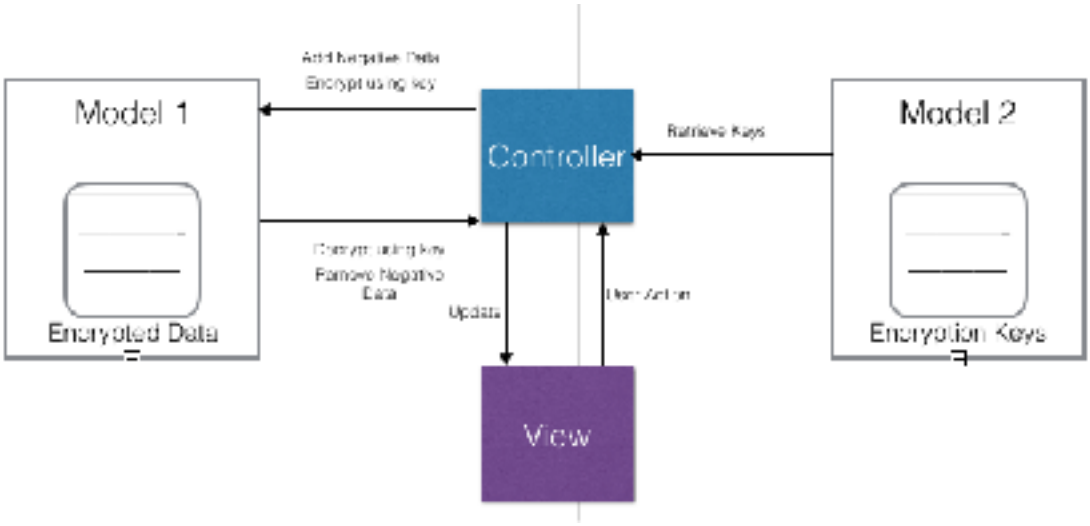

Figure 5: Flow Diagram of Encryption/Decryption process

If the user does not want to encrypt the data, we have added an option in the user interface. This option is available during a group creation or edition. Under Manage Groups section, you can create a new group where there is a drop down menu for Encryption field. This gives two options: ‘Enable’ to enable the encryption of data and ‘Disable’ to disable the encryption of data. By default, ‘Disable’ is selected.

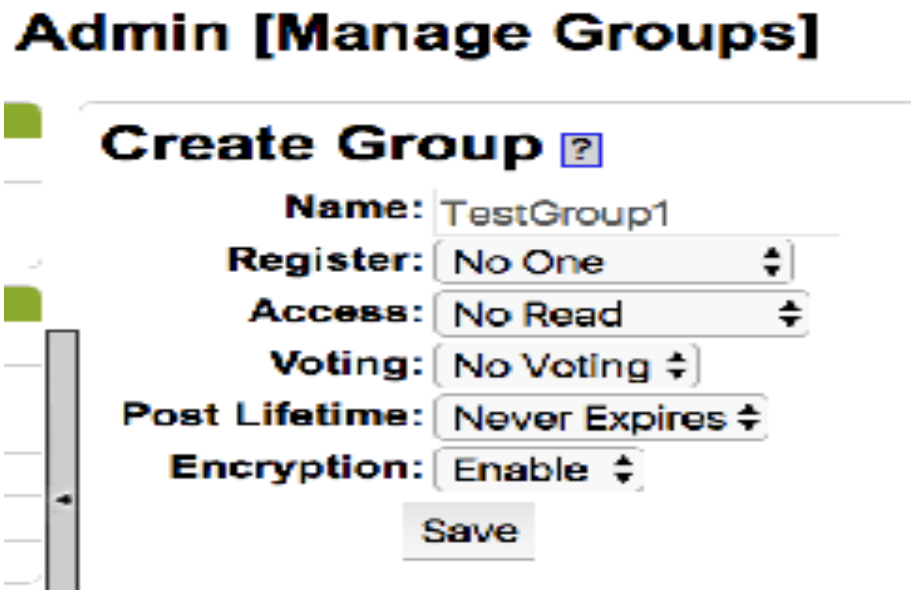

Figure 6: Create a new encrypted group

If encryption is enabled for a group, all posts in that group will be encrypted before storing it to the database. When displaying the posts of a group, the key which is stored in an external database is accessed first in order to decrypt the data before displaying it in the discussion board.

Figure 7: View encrypted posts in a group

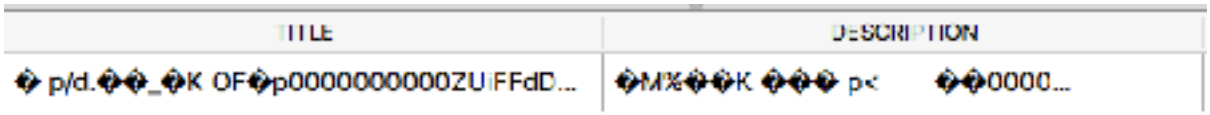

Figure 8: Encrypted values stored in a database

Adding Differential Privacy according to the defined policy on selected data

The main algorithm for adding Differential Privacy as described in the Background section is implemented in this section. The algorithm adds an **ε**-noise taken from an L_1 gaussian source to the actual value (Dwork, 2006).

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

The value is calculated by Integration method. Since function consists of absolute value, the integration is broken down into two to remove absolute from the function. First integral runs from 0 through actual value, and second integral runs from actual value through max_value. Then after using substitution rule, first and second integral range change, say [a,b] and [b,c].

Adding Differential Privacy in Group Statistics

The first implementation is done for the statistics of each group. Under each group, there are statistics that are displayed showing number of views in each group, thread and wiki. This information can be sensitive for a discussion board system as it can reveal which user has viewed which thread recently or in the past.

So, we added differential privacy to the group's statistics. For each time period under group, thread and wiki, we calculated the views based on the algorithm described by Dwork (2006) and displayed the calculated values in the summary page. One of the challenges here is to make sure the displayed values for each time period is logical. For example, let's say the algorithm calculated some value for the last hour of a group, say 8. Then if the algorithm calculated some value for the last day of that group, which is lower than previous time period (8), then we will have to make adjustments in the calculated value. Thus we have to make sure, value calculated by each time period is at least as large as it's lower time period.

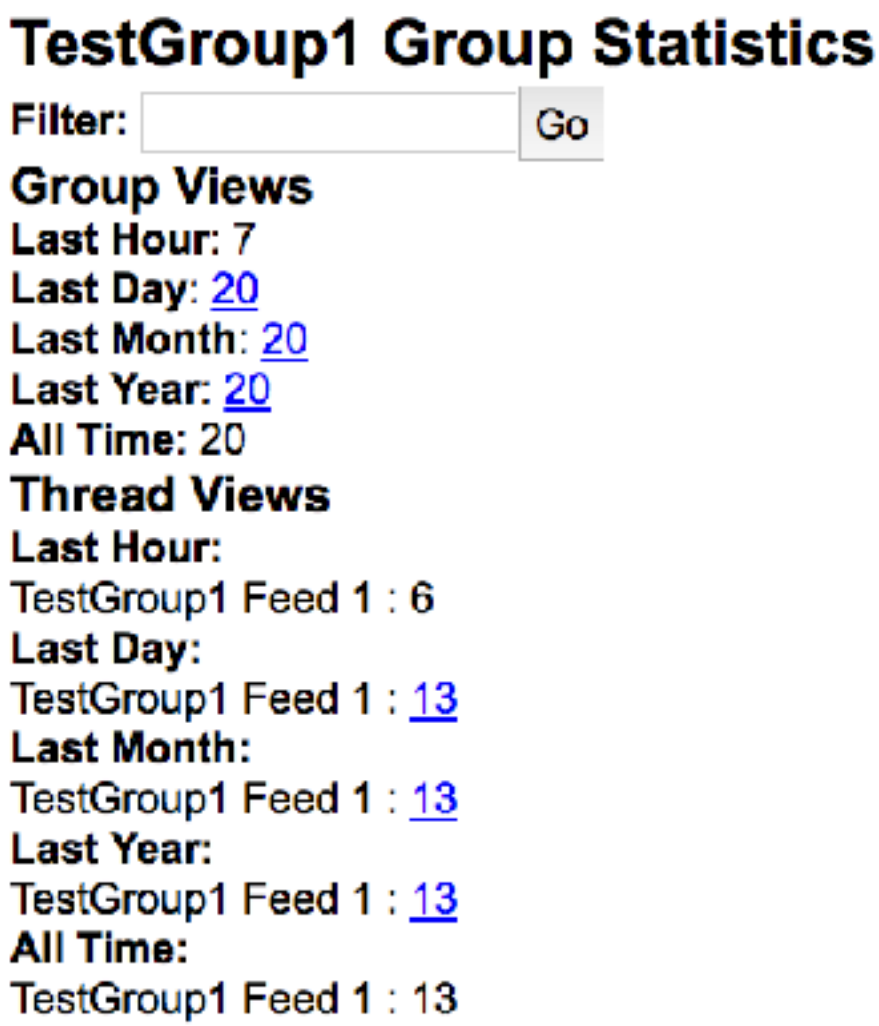

Figure 9: Groups statistics summary after adding differential privacy

Adding Differential Privacy in Search Statistics

The next use of differential privacy in Yioop we implemented was on the query statistics page. This is the page that displays the statistics about each query entered by a user in the search. A user uses the search tool to do a search by typing a single or multiple words. As this is a sensitive information about the user, it becomes critical to ensure the privacy of the user using the system. Thus, we are adding the privacy in the statistics generated from the user's queries so it becomes difficult for someone to find out any information about the user from the numbers displayed in the statistics.

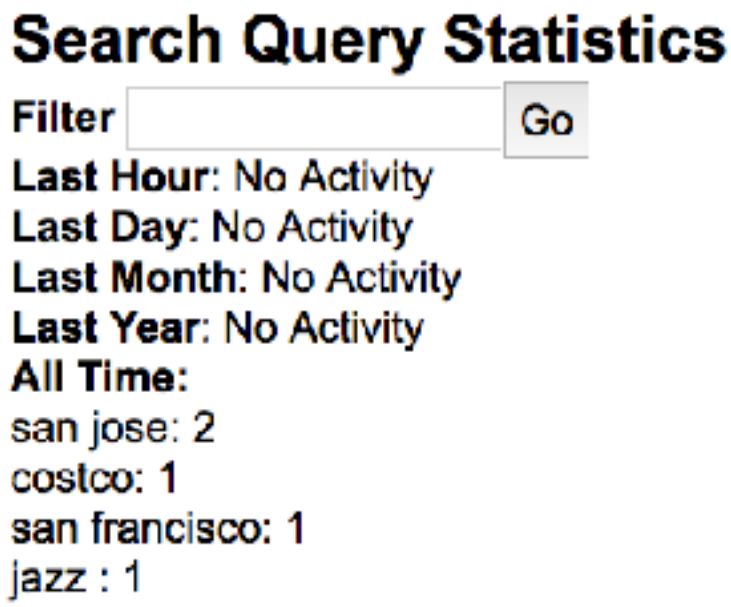

Figure 10: Query statistics before adding differential privacy

The figure belows displays the search query statistics after differential privacy has been enabled. This fuzzifies the actual count for each search query thus making it incomprehensible for anyone to extract the exact information.

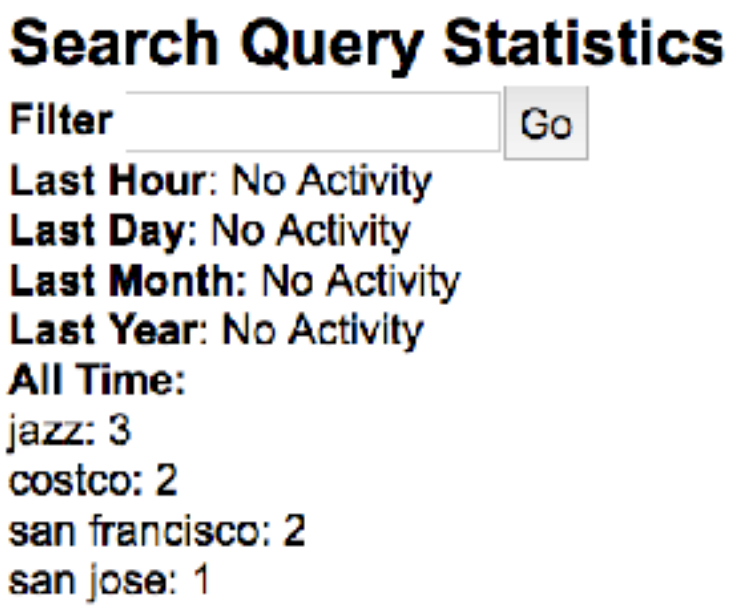

Figure 11: Query statistics after adding differential privacy

CHAPTER 6

TESTING AND EXPERIMENTS

This section discusses testing and experiments done for the new changes for adding Differential Privacy in the system. It is divided into following parts:

- Basic Setup
- Testing Differential Privacy in Group Statistics
- Testing Differential Privacy in Query Statistics
- Testing the performance when encryption is enabled

Basic Setup:

To test the way Differential Privacy is implemented on query and group statistics page, we first need to generate statistics. So, we first created a list of 100 users and 50 groups. Then we inserted groups for these users. Each user ‘i’ belongs to group ‘Groupi’. In order to test different scenarios, we are using Group1 as a reference group. So we are adding multiple users to this Group by assigning the first 50 users to Group1. Then under Group1, we created 20 different threads. With this basic setting, we can test the changes in the UI of Yioop for different scenarios.

$ php ManyThreadExperiment.php

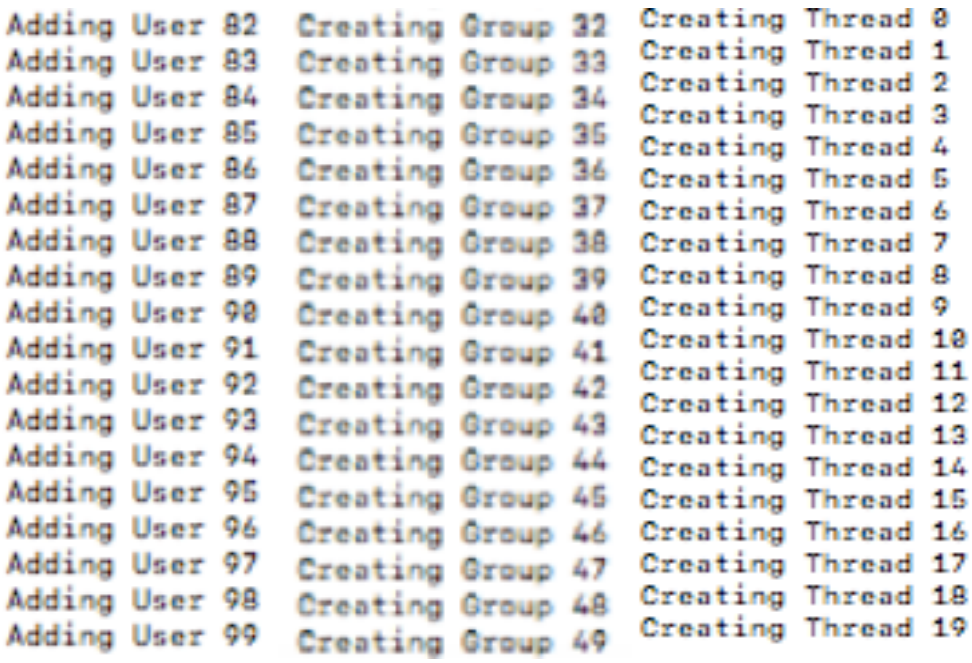

Figure 12: Experiment to setup users/groups/threads

Testing Differential Privacy in Group Statistics

We need data to generate statistics about the group in order to run the experiment for testing differential privacy in the group statistics. For this we need a simulation program that makes different users visit different threads. In order to simulate different users visiting different threads, we added a program that extends from the basic setup. This extension now simulates different users viewing each of the 20 threads. This way we collect statistics about each of the 20 threads in Group1 for different time periods (hourly, weekly, monthly, yearly).

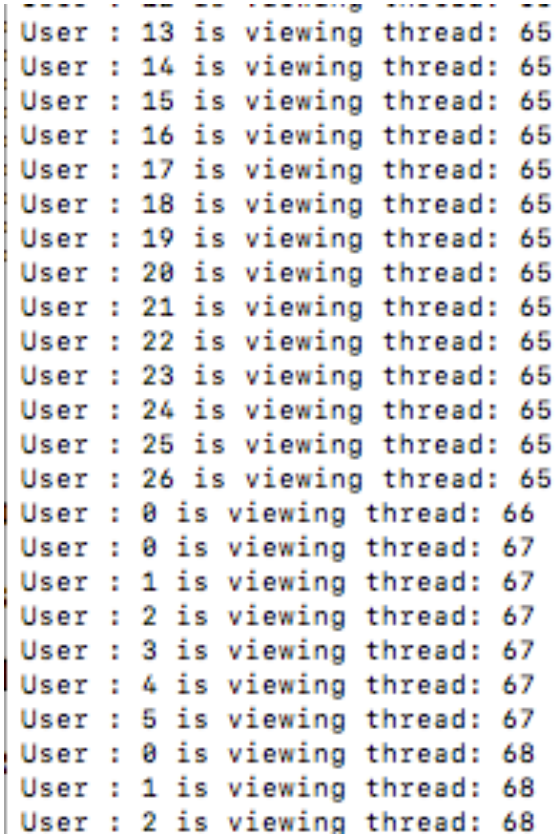

Figure 13: User experiment for viewing different threads

We then tested the UI by comparing the Statistics of the group when enabling Differential Privacy versus not enabling Differential Privacy.

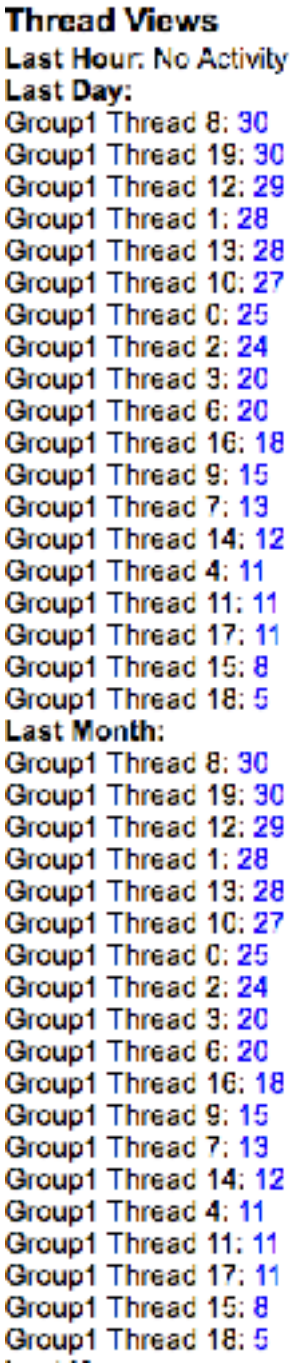

Figure 14: Experiment showing statistics before enabling differential privacy

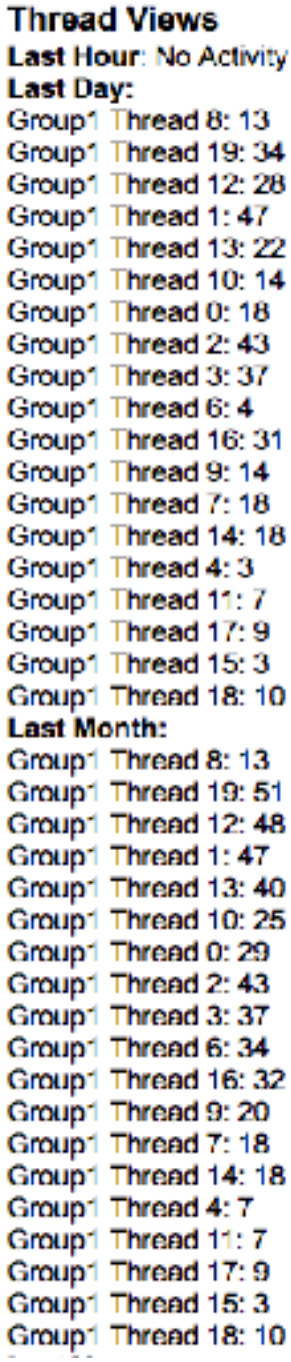

Figure 15: Experiment showing statistics after enabling differential privacy

By looking at the above two screenshots, the statistics displayed by differential privacy does not reveal exact count which is achieved by adding the gaussian noise to the actual value as described in the implementation section. Therefore, it becomes difficult for an adversary to perform statistical attacks against query and discussion board statistics.

Testing Differential Privacy in Query Statistics

To test differential privacy implemented in query statistics page, we first need to generate statistics of the queries. So we created list of search queries and performed searches for each query in random numbers. We then looked at the counts of each query in the statistics summary page and compared the statistics between non differential privacy setting and differential privacy setting. The counts in differential privacy setting have been deviated from the actual count as expected.

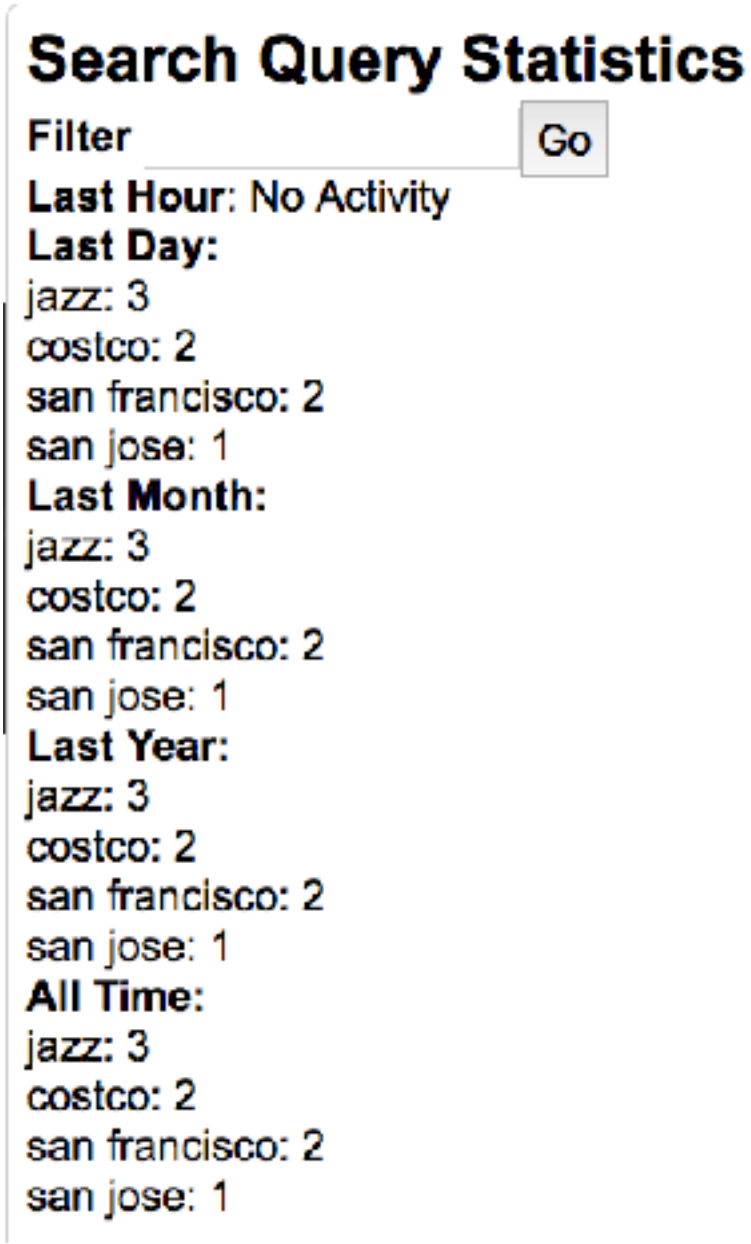

Figure 16: Experiment showing statistics after enabling differential privacy

Testing the performance when encryption is enabled

When encryption is enabled in the group, every threads posted in the group are encrypted. Since we are using column level and field level encryption, we are not encrypting the entire database but only some columns in the table.

For the testing, we used the basic setup described in ‘Basic Setup’ section but with one change when creating groups. Here we are passing an option to encrypt the group during group creation. So all the threads that are created are now encrypted in the database. We are creating the same 20 threads as used in ‘Testing Differential Privacy in Group Statistics’ section but these are now in encrypted mode.

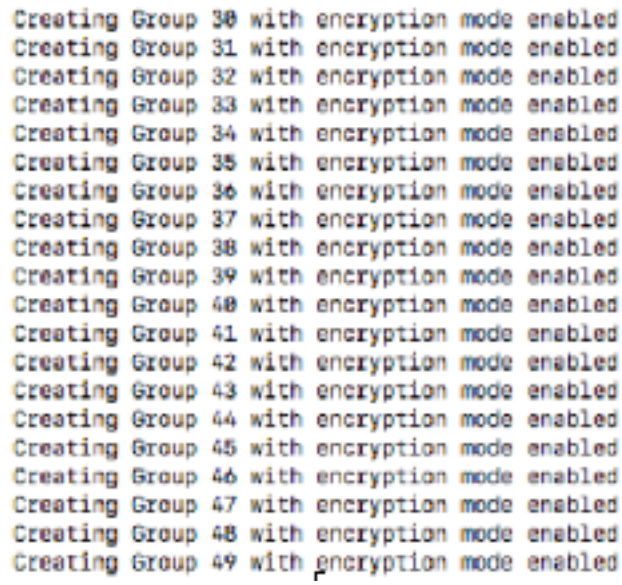

Figure 17: Creating groups with encryption enabled

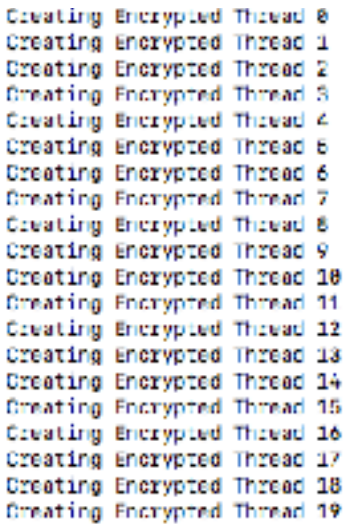

Figure 18: Creating threads in an encrypted group

Figure 19: Displaying threads in an encrypted group

Here the values stored in the database are in an encrypted mode and the keys needed to decrypt those data are stored in an external database.

5	11	H5[illegible]	b[illegible]	1493693342	1493693342
5	12	[illegible]	[illegible]	1493693342	1493693342
5	13	[illegible]	[illegible]	1493693342	1493693342

Figure 20: Encrypted values stored in a database

For our database, we are encrypting two columns of only one table and performing database operations such as insertion, deletions, queries, etc on those encrypted columns. For a moderate size of database, there is not much impact on the performance but as the size grows, it can impact the overall performance.

CHAPTER 7

CONCLUSION

In this project, we have implemented Differential Privacy as a privacy measure to some of the components in the Yioop search engine and open source discussion board system. As more and more attacks on the database systems are happening on a regular basis, it is extremely critical to protect the data from malicious users.

We are concerned not only with malicious users but even with whether users of the system with some negative intention can derive some sensitive data from non-sensitive statistical results displayed by the system. Differential privacy adds an additional layer of privacy to very sensitive data by adding some gaussian noise to the actual data and displaying the fuzzified data. Thus it makes anyone difficult to extract any information from the statistics displayed by the system.

We identified critical components of the system that are vulnerable to statistical attacks and applied the concept of differential privacy to those components. We mainly covered the statistics generated by the group and search analytics. These data can be very sensitive to an individual user. By enabling differential privacy, we are ensuring a user not to reveal any individual information whether or not the user stays or drops the participation in the database.

We also added another layer of protection to the database by encrypting the database at a column level. Instead of encrypting the entire database, we identified some of the critical

fields in the database that requires more protection and worked only in those fields for adding encryption such as title and description of a thread, its comments and replies, etc.

There are various techniques of adding privacy of the user participating in the statistical database. We explored few ways to achieve it. The importance of privacy-preserving statistical database is high as more and more users are using online platforms today and their data are used for many purposes by the companies.

REFERENCES

[1] Dwork, C. Differential Privacy, 33rd International Colloquium on Automata, Languages and Programming, part II, p.1-12, 2006.

[2] Patel, A., Sharma N., Eirinaki M., Negative Database for Data Security, Proceedings of the 2009 International Conference on Computing, Engineering and Information, p.67-70, April 02-04, 2009.

[3] Database Encryption, Retrieved May 03, 2017 from https://en.wikipedia.org/wiki/Database_encryption.

[4] Burtescu, E. (2009). Database Security Attacks and Control Methods, Journal of Applied Quantitative Methods, vol. 4, no. 4, June 2009.

[5] Kamaraju, A.(February 9, 2012), Database encryption demystified: Four common misconceptions. Retrieved May 03, 2017 from http://www.zdnet.com/article/database-encryption-demystified-four-common-misconceptions/.

[6] Dwork, C. and Roth, A. The Algorithmic Foundations of Differential Privacy, Foundations and Trends in Theoretical Computer Science Vol. 9, Nos. 34 (2014) 211 407, 2014.

[7] Dwork, C, The Promise of Differential Privacy: A Tutorial on Algorithmic Techniques, Proceedings of the 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science, p.1-2, October 22-25, 2011 [doi>10.1109/FOCS.2011.88].

[8] Differential Privacy. Retrieved May 17, 2017 from https://en.wikipedia.org/wiki/Differential_privacy.