*Florida International University*

*School of Computing and Information Sciences*

Software Engineering Focus

Final Deliverable

Security Evaluation of Encrypted Databases

(SEED v1.0):

**Team Members:** Steven Caceres, Joannier Pinales

**Product Owner(s)**: Mireya Jurado

**Mentor(s)**: Mark Finlayson, Rob Cunningham

**Instructor**: Masoud Sadjadi

The MIT License (MIT)

Copyright (c) *2016 Florida International University*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON INFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

***Abstract***

*This document presents the information necessary to gain a good understanding of the SEED project. The purpose of this project was to evaluate, and make more secure, encrypted database systems. The system evaluated throughout this project was CryptDB, an encrypted proxy database system developed at MIT that uses multiple layers, and methods of encryption. In order to make the system more secure, attacks were conducted in order to find a data leak or vulnerability in the system. After conducting frequency analysis on columns containing deterministically encrypted data, the frequency distribution for that column was obtained, and the plaintext values were inferred using an auxiliary data set. Once this data leak was found, a countermeasure was designed and implemented. By controlling the insertion and deletion of fake data into the database, the system is able to keep the frequency distribution of the data uniform. The results of the countermeasure prevents frequency analysis from being done on the system, and was implemented with minimal alteration to the system itself.*

**Table of Contents**

**Introduction**

Current System ……….……………………………………………………………………………………………………………… 5

Evaluating System Vulnerabilities ………………………………………………………………………………………………. 5

Purpose of New System …………………………………………………..………………………………………………………… 6

**User Stories**

Implemented User Stories ………………………………………….…………………………………………………....……….. 7

Pending User Stories …………………………………………………………………...……………………………………...….. 12

**Project Plan**

Software Resources ……………………………………………………...……………………………………..……………….….14

Sprints Plan ……………………………………………………………..…………………………………………..………………. 15

*Sprint 1*  …………………………………………………………………………………………………………………………... 15

*Sprint 2*  …………………………………………………………………………………………………………………………... 15

*Sprint 3*  …………………………………………………………………………………………………………………………... 16

*Sprint 4*  …………………………………………………………………………………………………………………………... 17

*Sprint 5*  …………………………………………………………………………………………………………………………... 17

*Sprint 6*  …………………………………………………………………………………………………………………………... 18

**System Design**

Architectural Patterns …………………………………………………………………………………...……………………... 19

System and Subsystem Decomposition …………………………………………………………………………………………. 20

Deployment Diagram …………………………………………………………………………………………………….………… 21

Design Patterns ………………………………………………………………………………………………………….…………. 21

**System Validation**  …………………………………………………………………………………………………………………….22

**Glossary**  ………………………………………………………………………………………………………………………………….27

**Appendix**  ………………………………………………………………………………………………………………………………….28

Appendix A - UML Diagrams ……………………………………………………………………………………………………. 28

*Static UML Diagrams*  ……………………………………………………………………………………………………….38

*Dynamic UML Diagrams*  …………………………………………………………………………………………………..40

Appendix B - User Interface Design …………………………………………………………………………….……………… 44

Appendix C - Sprint Review Reports …………………………………………………………………………...……………… 45

Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents …………………………………………………………………….…………… 48

**References** ……………………………………………………………………………………………………………………...………...54

# Introduction

## Current System

The current system, CryptDB, is an encryption proxy system that uses multiple layers and methods of encryption in order to secure its data but still allows operations to be performed against it. Used in unison with the MySQL client and server applications, it takes advantage of MySQL proxy to alter the queries and result sets being sent from the client and server respectively. The main purpose of the system is to store only encrypted data on the server, while allowing the client to query the encrypted data at which points it gets decrypted and sent back. The idea behind encrypting the data before it is stored comes from a threat model where the machine where the database resides gets breached. In any regular database, when this happens the data is completely stolen, and there’s little that can be done to prevent it. In the case that CryptDB was being used, in such situations, the data is encrypted and secured to an extent.

Although the CryptDB system takes away some functionality from the regular database systems, it still allows for a wide range of functionality while also keeping the data more secure. In order to accomplish such security, multiple methods of encryption are utilized, most notably randomized, homomorphic, word search, and deterministic encryption. Random encryption gives the highest security guarantee, allowing zero leakage of information about the data. Homomorphic encryption allows calculations to be performed with the encrypted data. Deterministic encryption allow comparison between any two encrypted values. Furthermore, multiple layers of these encryptions are used, referred to as onions, on the columns of data depending on their type. By combining these layers, CryptDB is able support different types of queries by using the layer that supports the operation requested. When a query is received from the client, layers are peeled from the onion, if necessary, to perform the query.

## Evaluating System Vulnerabilities

Aspiring to create perfect systems, could be a software engineer’s greatest weapon, but assuming the system is already perfect the greatest downfall. Even though CryptDB is a very powerful system that prevents data being stolen by any average attacker, there are claims in recent research studies that have exposed some of its vulnerabilities. Recent assessments of the system have proven effective ways to obtained the values of the encrypted information under certain attacks using inference attack. Some of the attacks used to expose CryptDB vulnerabilities are: frequency analysis, a well know attack that decrypts deterministic encryption based on an auxiliary dataset[6], l2 optimization, a family of attacks introduced by Naveed kamara to also decrypt deterministic encryption, based on combinatorial optimization techniques [6]. The success of these two inference attacks depends on the auxiliary data set used to obtain information about the data. In the recent study by Naveed, he was able to recover a percentage of the encrypted data in multiple columns. These successful are the reason behind this project, to first evaluate and recreate each of attack, then implement countermeasures in the CryptDB system to prevent them from recurring.

## Purpose of New System

One of the many advantages of the CryptDB system is the ability to maintain a wide range of functionality by using different types of encryption in different layers. Only peeling off the necessary layers when a query is requested that requires a specific encryption. One of the drawbacks to this, is that some forms of encryption leak data, as stated before. Encryptions such as deterministic and order-preserving can be exploited to recover encrypted data, the new system aims to stop data leaks. Specifically, the new system focuses on frequency analysis inference attacks and aims to deter it by keeping the distribution of data within the columns uniform.

To accomplish this, the frequency of data is kept track of throughout the use of the system. Whenever data is inserted into a table, the count of that data is increased in the frequency tracker. Fake data is only inserted into columns of lesser count when an insert statement that would imbalance the frequency distribution is detected. By keeping the distribution uniform throughout the system, if an attacker that has access to the encrypted values were to run a frequency analysis, they would not be able to correlate the distribution with any auxiliary set that they may have.

## 

# 

# User Stories

The following section provides the detailed user stories that were implemented in this iteration of the Security Evaluation of Encrypted Databases project. These user stories served as the basis for the implementation of the project’s features. This section also shows the user stories that are to be considered for future development.

## Implemented User Stories

**#668 - Familiarize yourself with CryptDB:**

* Description:
  + As a researcher, I would like to run basic queries with CryptDB, so that I may familiarize myself with it.
* Acceptance Criteria:
  + Run basic SQL commands such as CREATE, DROP, INSERT, DELETE, SELECT, UPDATE within CryptDB.

**#669 - Import data into the database:**

* Description:
  + As a researcher, I would like to data in the database, so that I may run tests on it.
* Acceptance Criteria:
  + Import a large (1,000-10,000 rows) data set into the database.
  + I would like to do this using a script.

**#673 - Generate data:**

* Description:
  + As a researcher, I would like to generate data with a desired distribution, so that I may import it into the database and then test frequency analysis against it.
* Acceptance Criteria:
  + Program that creates a .csv file of data that can be set to distributions of my choosing

**#675 - Set up Docker:**

* Description:
  + As a researcher I would like to have the CryptDB system self contained, and easy to set up.
* Acceptance Criteria
  + CryptDB running inside a docker container, with scripts auto setting up all necessary dependencies

**#670 - Analyze the differences:**

* Description:
  + As a researcher, I would like to know the differences between the two versions of CryptDB, so that I may accurately analyze the data in the database.
* Acceptance Criteria:
  + Run a diff on both repositories, note any major changes to the code.

**#671 - Research the different encryptions being used:**

* Description:
  + As a researcher, I would like to learn more about the different types of encryptions being used, so that I may know how to approach them.
* Acceptance Criteria:
  + Look into what encryptions CryptDB utilizes
  + Search for a way to best approach these types of encryptions

**#672 - Count the Frequency of Data in a Column:**

* Description:
  + As a researcher, I would like to be able to count the frequency of data in a column, so that I may generate percentages and distributions for comparison.
* Acceptance Criteria:
  + Outputs frequency of data from a chosen attribute in a table.

**#674 - Instal Onion Layer Web View:**

* Description:
  + As researchers, we would like to see the onion layer web view that allow us to see the different onion layers per column
* Acceptance Criteria:
  + Onion web view running, able to be accessed from the browser

**#676 - Import Data Using Sensitive Annotations:**

* Description:
  + As researchers we want to make sure the data is inserted correctly into CryptDB to make sure we dont make the same mistake as Naveed
* Acceptance Criteria:
  + Data inserted with the columns using the CryptDB sensitive annotation

**#677 - Re-install CryptDB from the original repo:**

* Description:
  + As a researcher/developer we would like to recreate the attack using the original database system from MIT.
* Acceptance Criteria:
  + Original CryptDB installed

**#678 - Inserting data with annotations**

* Description:
  + As a researcher/developer I would like test the different types of attacks against the sensitive annotations to see if it’s possible to recover any encrypted data using frequency analysis.
* Acceptance Criteria:
  + Database schema with annotations

**#679 - Debug Onion Web View**

* Description:
  + As a developer/researcher we would like to see a web view of the different onions in each columns. This already exist but it’s not working properly
* Acceptance Criteria:
  + Onion Web view properly working

**#680 - Re-run the frequency analysis on the original CryptDB:**

* Description:
  + As researcher/developers we would like to know if attacks are still possible in the original CryptDB.
* Acceptance Criteria:
  + Successful frequency analysis attack.

**#681 - Research area to pivot the project towards implementation:**

* Description:
  + As a student, I would like to research an area to pivot the project towards implementation
* Acceptance Criteria:
  + Different areas which we can change the project towards.

**#682 - Research Usefulness of sensitive annotations**

* Description:
  + As a database manager I would like to see how useful are the sensitive annotations to the overall security of the database so that I can decide which columns I should make sensitive when creating tables
* Acceptance Criteria:
  + Detailed explanation of the annotations and its usefulness

**#683 - Research Possibility of fake data insertion**

* Description:
  + As a database manager I would like to see how can we prevent attackers from performing frequency analysis attacks and at what cost so that I can see what performance overhead this extra security measure will have.
* Acceptance Criteria:
  + Detailed explanation of everything it takes to make fake data insertion to the database.

**#684 - Research usefulness of the onion web view tool:**

* Description:
  + As a researcher/developer I would like to know how useful the web view tool is as a possible candidate to pivot our project towards.
* Acceptance Criteria:
  + A detailed view of the web view and its usefulness.
  + What can we add to it if it proves useful.

**#685 - Parse query to get the frequencies being inserted**

* Description:
  + As a DBMS I would like have the database protected against frequency attacks so that I can ensure the safety of my information to the customers. Parsing the insertion query to extract frequencies will impact performance tremendously by not having to read these values from the database.
* Acceptance Criteria:
  + All insertion queries parsed
  + Frequency for all data items stored in a file

**#686 - Generate fake data and add to insertion query**

* Description:
  + As a DBMS I would like to insert fake data into the database so that if encrypted information gets stolen it does not get decrypted by frequency analysis. Calculate how much fake data should be inserted and for which value by using the flat histogram algorithm.
* Acceptance Criteria:
  + Fake data added to the insertion query according the difference in frequency
  + Fake data inserted into the database

**#687 - Create extra column for fake flag whenever a table is created:**

* Description:
  + As a DBA, I would like the system to create a fake column whenever a table is created so that i can mark the different types of data.
* Acceptance Criteria:
  + Each create table query becomes modified to add a fake column to the table.

**#688 - Filter fake results from queries:**

* Description:
  + As a DBA, I would like to filter out the fake data so that only real data is returned to me. This must be done in the server without using any where clauses.
* Acceptance Criteria:
  + Result set should contain only real data.
  + Filtering done properly in the proxy only.

**#689 - Store Frequencies in a file**

* Description:
  + As a DBMS I would lik**e** to have the frequencies saved on a file so that when the database restarts we have an accurate count.
* Acceptance Criteria:
  + Frequencies saved in a file
  + Frequencies loaded from the file at startup

**#691 - Omit the fake attribute from the result set:**

* Description:
  + As a user, I would like the fake column to be omitted from the result set, so that I do not have an unnecessary column causing confusion.
* Acceptance Criteria:
  + When the result set is returned to the user, there should not be a fake attribute

**#692 - Create a basic web app to use with CryptDB:**

* Description:
  + As a user, I would like to have a web app to use with CryptDB so that I can easily view, and alter my data.
* Acceptance Criteria:
  + Ability to sign in
  + Able to view databases, tables, and their values
  + Able to create and drop databases and tables within the web app

## 

## Pending User Stories

**# 690 - Parse Remove Query and Modify to update**

* Description:
  + As a DBA I would like the remove queries to not change the flat histogram so that the database remains secure at all times .
* Acceptance Criteria:
  + Remove queries parsed
  + Remove queries modify to update fake = 1 instead

# Project Plan

This section describes the planning that went into the realization of this project. This project incorporated the agile development techniques and as such required the sprints to be planned. These sprint plannings are detailed in the section. This section also describes the components, both software and hardware, chosen for this project.

## 

## Software Resources

The following is a list of all software resources that were used in this project:

**Software**

* **MySQL Client/Server:** MySQL is an Open Source SQL database management system. MySQL is a very popular relational database that stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed[1]
* **MySQL Proxy:** MySQL proxy is a simple program that sits between the client and MySQL server(s). By using it the app can monitor, analyze or transform the communications, very useful for load balancing and security.
* **Docker Container:** A container platform is complete solution that allows organizations to solve multiple problems across a diverse set of requirements[2]. A Docker container is a lightweight virtual machine that is very powerful in terms of scalability and self containment. By using docker, developer projects can be spinned up really fast and easy. In terms of this project, by using docker all the attacks were performed in a self contained environment.
* **Virtual Machine (Ubuntu 14.04):** A virtual machine (VM) is an emulation of a computer system. Virtual machines are based on computer architectures and provide functionality of a physical computer[3]. This project used Ubuntu 14.04 virtual machine, instead of the latest 16+ versions. This was because the CryptDB system was developed in Ubuntu 12 so the closest compatible version was 14.04. It was possible to also run CryptDB in Ubuntu 16 but some features did not work.
* **Lua:** Lua is a programming language that can be easily embedded with C and C++, however its garbage collection property make it very powerful.
* **Python:** A very simple but powerful programming language
* **PHP:** Scripting language mostly used in servers.

## Sprints Plan

#### Sprint 1

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Set Up CryptDB
* Migrate CryptDB to Ubuntu 16.02
* Insert Data Properly

The team members indicated their willingness to work on the following user stories.

Steven Caceres and Joannier Pinales

* Set up CryptDB Up and running
* Migrate CryptDB to Ubuntu 16.02
* Insert Data into CryptDB

#### Sprint 2

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Run diff between practical CryptDB and the original CryptDB and analyse the differences
* Insert data properly intro CryptDB using the sensitive annotations
* Read more into OPE encryption and DET encryption, and other encryptions used by CryptDB
* Create script to run frequency analysis in the columns
* Setup onion web view

The team members indicated their willingness to work on the following user stories.

Steven Caceres

* Run diff and analyse the differences #670
* Research the different encryptions being used, specially OPE and DET #671
* Create script to analyse frequencies #672

Joannier Pinales

* Import data using the sensitive annotation #676
* Install onion layer web view #674
* Research the different encryptions being used, specially OPE and DET #671

#### Sprint 3

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Get annotations to works
* Get onion Webview working
* Re run the attack with annotations
* Re install CryptDB from the original repo

The team members indicated their willingness to work on the following user stories.

Steven Caceres

* Rerun frequency attack #680
* Reinstall the original CryptDB - #677

Joannier Pinales

* Import data using the sensitive annotation on the original #678
* Onion Webview working properly - #679

#### Sprint 4

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Research the different areas we can take the project to in terms of implementation
* Look into the usefulness of these:
  + Onion web view
  + Sensitive Annotations
  + Fake data insertion

The team members indicated their willingness to work on the following user stories.

Steven Caceres

* Research areas to pivot the project - #681
* Research usefulness of onion web view - #684

Joannier Pinales

* Research usefulness of sensitive annotations #682
* Research usefulness of fake data insertion as a countermeasure #683

#### Sprint 5

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Fake data insertion for countermeasure against frequency analysis
* Save frequency in the file
* Filter out results based on schema
* Parse insert query

The team members indicated their willingness to work on the following user stories.

Steven Caceres

* Autocreate and store fake data column - #687
* Filter out fake results for the final result set - #688

Joannier Pinales

* Parse query in CryptDB to see the data that is being inserted #685
* Adjust query to insert fake data according to frequency proportions #686

#### Sprint 6

After discussion, the velocity of the team were estimated to be 4h a day.

The product owner chose the following user stories to be done during the next sprint. They are ordered based on their priority.

* Store frequencies in the a file and encrypt the file
* Intercept remove queries and modifying them update fake = 1
* Omit the fake attribute from the result set
* Create a basic web app to use with CryptDB

The team members indicated their willingness to work on the following user stories.

Steven Caceres

* Omit fake column from the results by deleting it from the result set table - #691
* Create a basic web app to make viewing the databases easier - #692

Joannier Pinales

* Store frequency in the file and encrypt file #689
* Intercept remove query and modify to update fake = 1 #690

# 

# System Design

This section contains information on the design decisions that went into this project. The architecture patterns are outlined and explained. The entire system is shown in a package diagram and the subsystems are explained. Finally, the design patterns used in the project are discussed.

## Architectural Patterns

## 

The architectural pattern of the system consists of a proxy and a database server with specified user defined functions to perform operations on the encrypted data. The proxy accepts all the client connections and intercepts all the queries issued then modifies them depending on which type of operation is being requested. The proxy then gives the database server the necessary encryption and decryption keys to perform the requested operation. The proxy also keeps track of which columns are sensitive and the frequency of each value in a column.

## System and Subsystem Decomposition

## 

## The subsystem consists of internal functions within the CryptDB system that handle query parsing, query rewrites, sensitive annotation schemas, encryption and decryption keys setup, database connection, thread handlers and meta tables handlers. The sub system handles a many required operations such as validating the queries passed from MySQL proxy, parsing it, rewriting it if necessary, it also calls the necessary SQL udfs that perform operations with the encrypted data.

## 

## Deployment Diagram

The deployment architecture consists of a single Docker container running both the proxy and MySQL server. Docker runs in any regular machine, and even though it’s self contained it exposes different ports to the machine that can be accessed by the client. In this case port 3307 is exposed by the container to wait for client connections. If the CryptDB system was to be deployed on a production environment the recommendation is run proxy in separate machines, to avoid a single point of hacker breach.

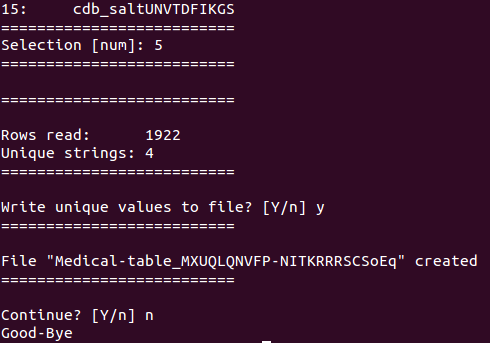
## 

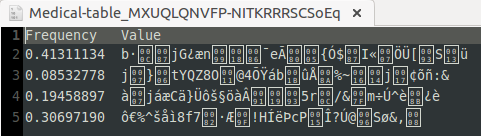
## Design Patterns

The main design pattern for the proxy system is wrapper design. MySQL proxy uses a Lua wrapper to communicate with CryptDB C++ functions. This allows MySQL proxy and CryptDB functionality to be completely decoupled, that is CryptDB’s module is completely independent from MySQL proxy. In the case where MySQL proxy updates its functionalities, this will not affect the CryptDB module. The CryptDB module just recieves queries and passes results from and to mysql proxy who then sends it to the clients. This also allows the developer to completely focus on CryptDB functionality and not worry about handling client connections and auth. Since the code for MySQL proxy is maintained by some other organization the wrapper pattern fits perfect into this architecture. In terms of the CryptDB module, it uses a composite design pattern mixed with a singleton connection module that communicates with MySQL server. The composite design in the main class aggregates the different subclasses, which also implement other subclasses , creating a hierarchy.

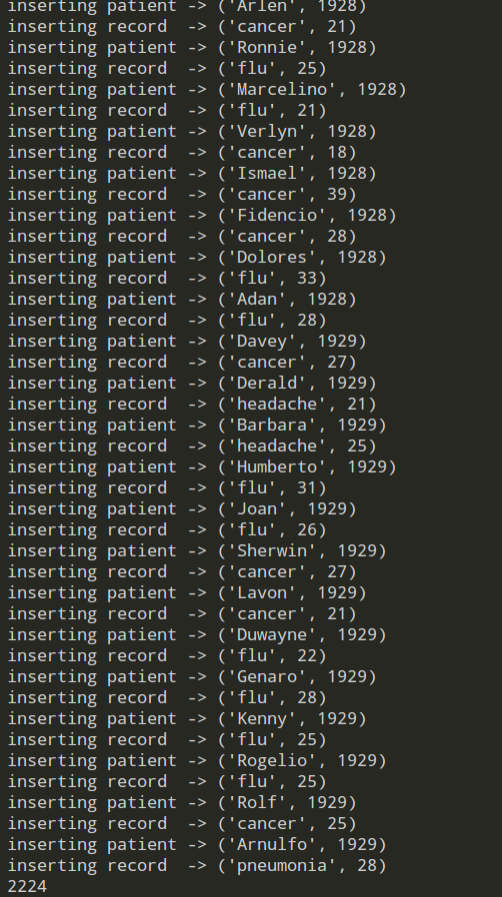
# System Validation

## #672 - Frequency analysis of chosen columns



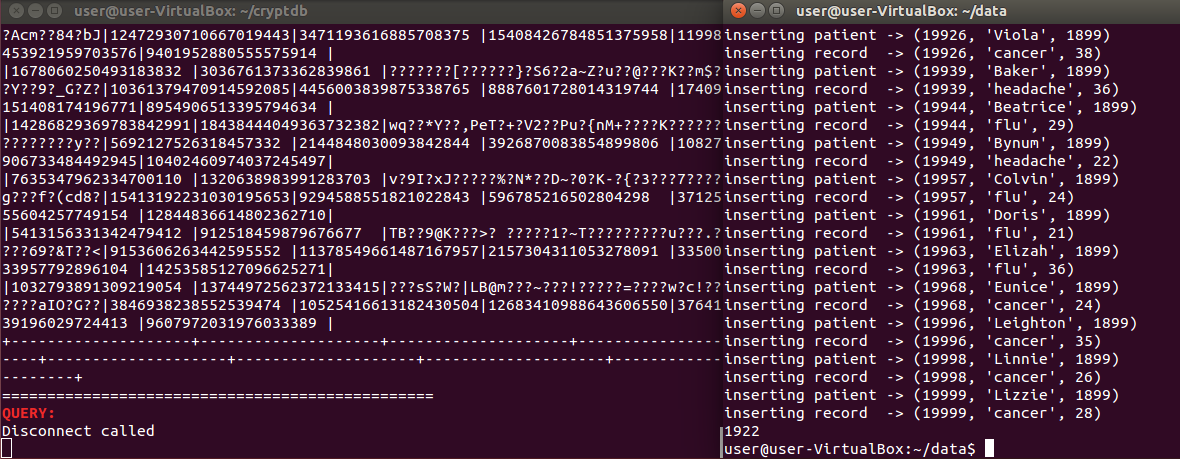


## #673 - Data generation with desired frequencies of 4 distinct illnesses

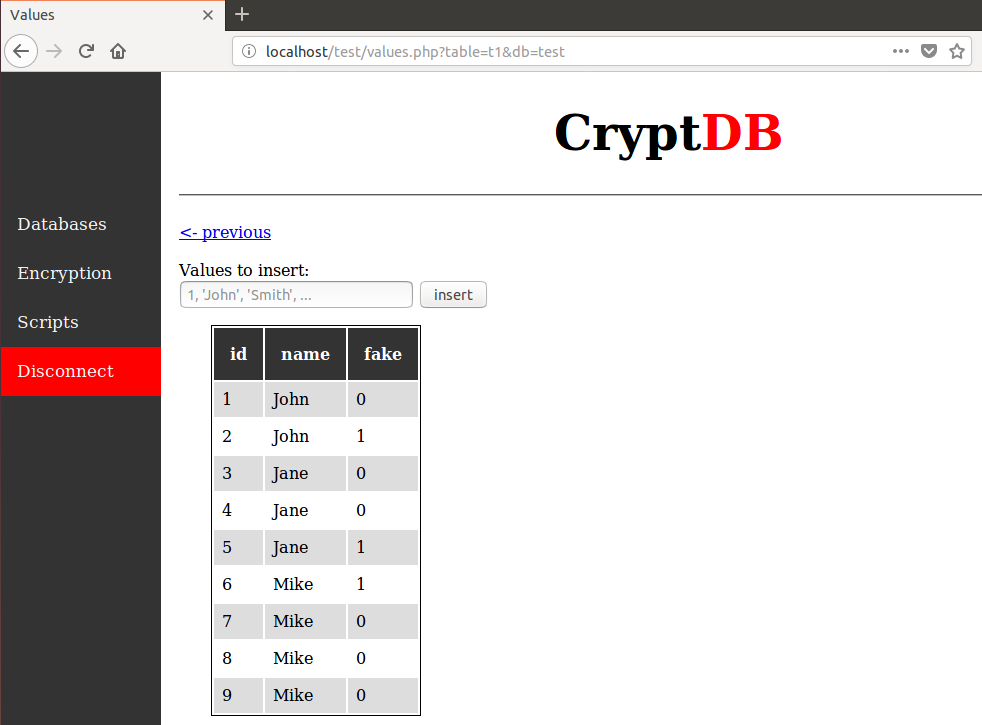


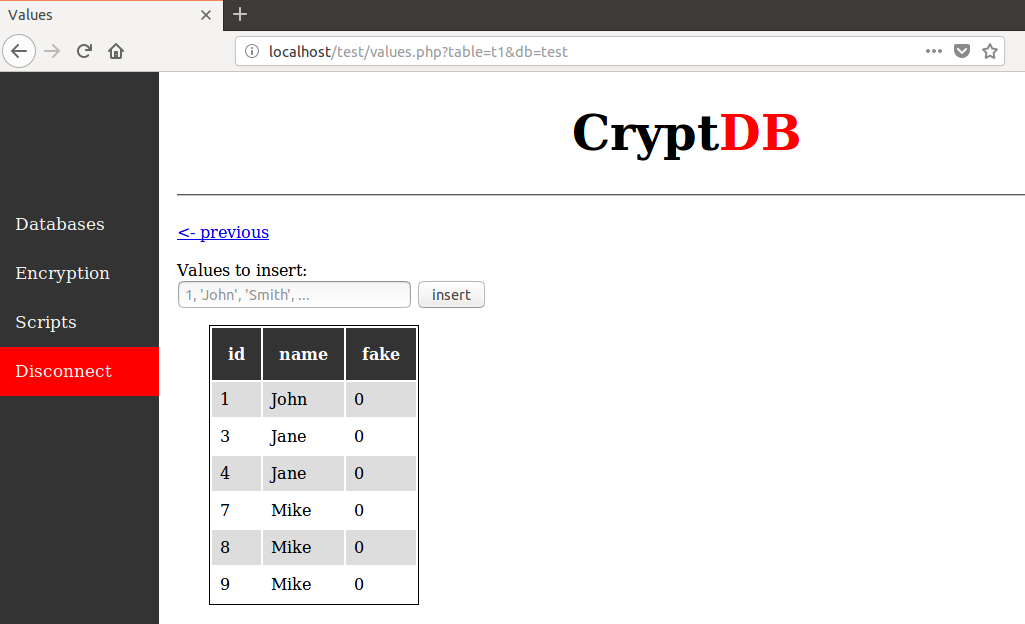
## #685 - Insert query parsed and frequency adjusted using flat histogram algorithmx

## #676 - Insertion of large amounts of data into CryptDB

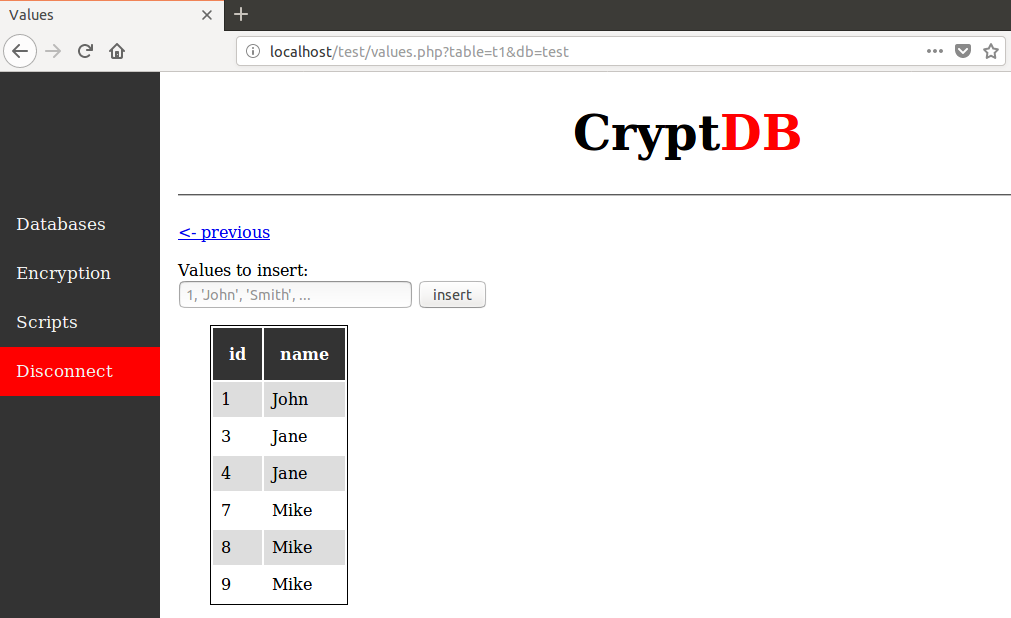


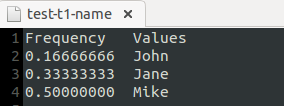
## #688 - Filter fake data from result set outside of the server



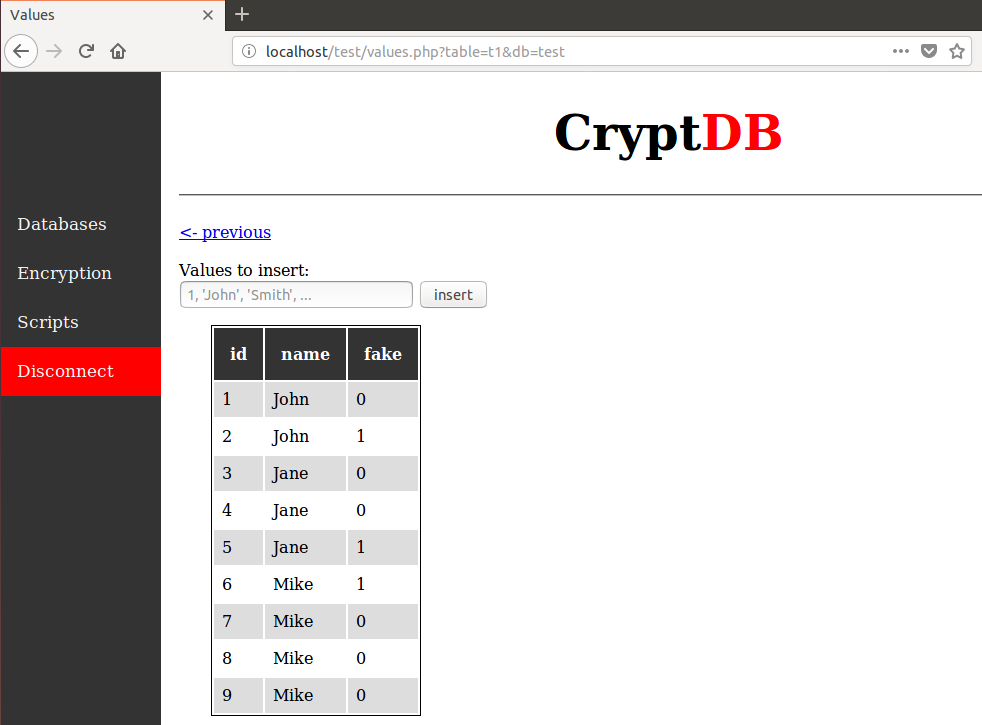


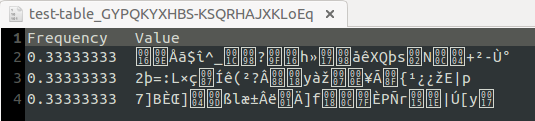
Data that the user has entered, and the resulting distribution of frequency analysis on the column





Data stored on the server, including the fake data that has been inserted, and the resulting distribution that an attacker would get if they were to run frequency analysis on the column.



****

# 

# Glossary

**Proxy Server:**  a machine that intercepts all communications between the actual server and the client.

**Inference Attacks:** data mining technique performed by analyzing data in order to illegitimately gain knowledge about a subject or database[7], in this case using an auxiliary database.

**Random (RND) Encryption**: provides the maximum security in CryptDB: indistinguishability under an adaptive chosen-plaintext attack (IND-CPA); the scheme is probabilistic, meaning that two equal values are mapped to different ciphertexts with overwhelming probability. On the other hand, RND does not allow any computation to be performed efficiently on the ciphertext. An efficient construction of RND is to use a block cipher like AES or Blowfish in CBC mode together with a random initialization vector (IV). (CryptDB mostly uses AES, except for integer values, where Blowfish is used for its 64-bit block size because the 128-bit block size of AES would cause the ciphertext to be significantly longer)[3].

**Deterministic (DET) Encryption**: encryption that has a slightly weaker guarantee than random, yet it still provides strong security: it leaks only which encrypted values correspond to the same data value, by deterministically generating the same ciphertext for the same plaintext. This encryption layer allows the server to perform equality checks, which means it can perform selects with equality predicates, equality joins, GROUP BY, COUNT, DISTINCT, etc.[3]

**Homomorphic (HOM) Encryption:** secure probabilistic encryption scheme (IND-CPA secure), allowing the server to perform computations on encrypted data with the final result decrypted at the proxy[3]. While fully homomorphic encryption is prohibitively slow [9], homomorphic encryption for specific operations is efficient. To support summation, we implemented the Paillier cryptosystem [10].

**Word Search (SEARCH) Encryption:** used to perform searches on encrypted text to support operations such as MySQL LIKE operator. Cryptdb implements the cryptographic protocol of Song et al. [11], which was not previously implemented by the authors; it also uses their protocol in a different way, which results in better security guarantees[3]. For each column needing SEARCH, the text is splitted into keywords using standard delimiters (or using a special keyword extraction function specified by the schema developer). Then repetitions in these words are removed, positions of the words randomly permuted and then word encrypted using Song et al.’s scheme, padding each word to the same size.

# Appendix

## Appendix A - UML Diagrams

### #688 Filter Fake Results

#### Use Case Diagram

### 

#### Sequence Diagram

#### 

### #687 Create Extra Fake Column Flag

#### Use Case Diagram

#### 

### 

### 

### 

### 

### 

### 

### 

### #686 Generate Fake Data and Insert Using Flat Histogram Algorithm

#### Use Case Diagram

#### 

#### 

#### Sequence Diagram

#### 

#### 

#### 

#### 

#### 

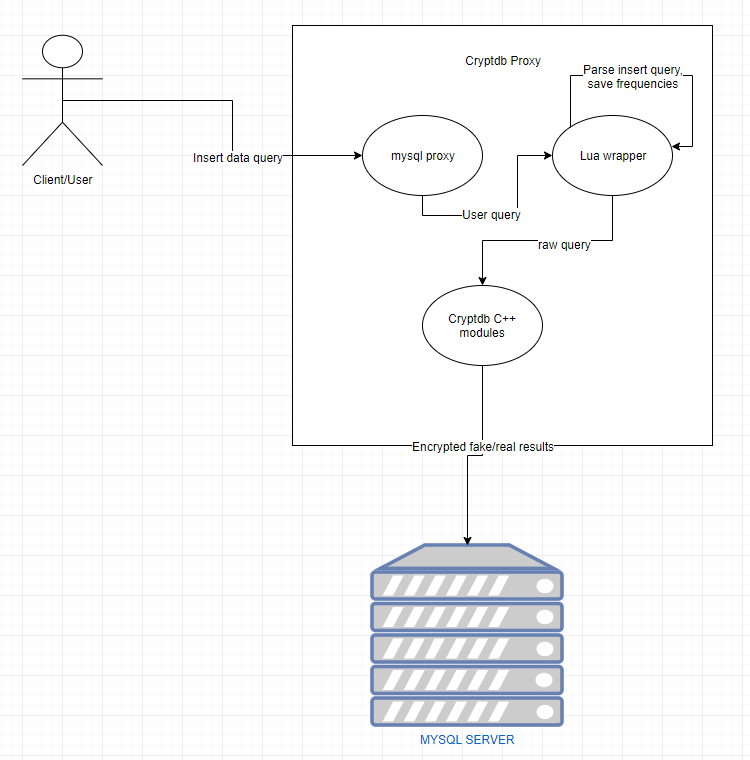
#### 

#### Class Diagram

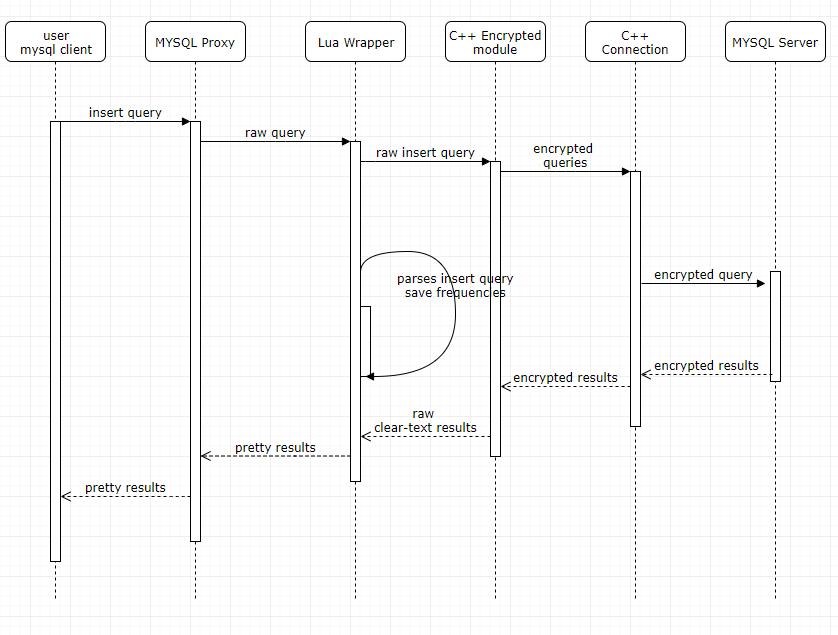
#### 

### #685 Read, Parse Query and Extract Frequencies

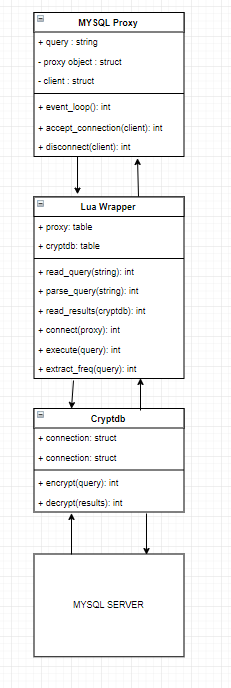
#### Case Diagram

****

#### Sequence Diagram

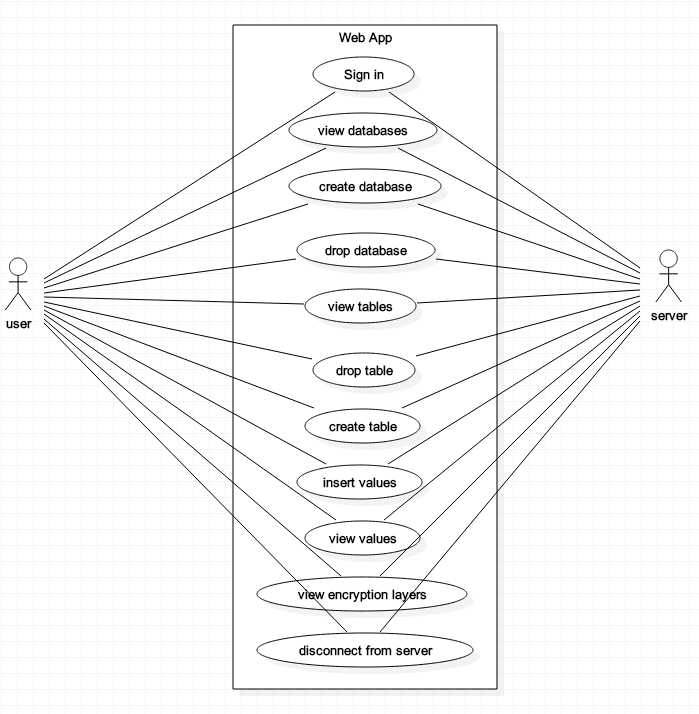
****

#### Class Diagram

****

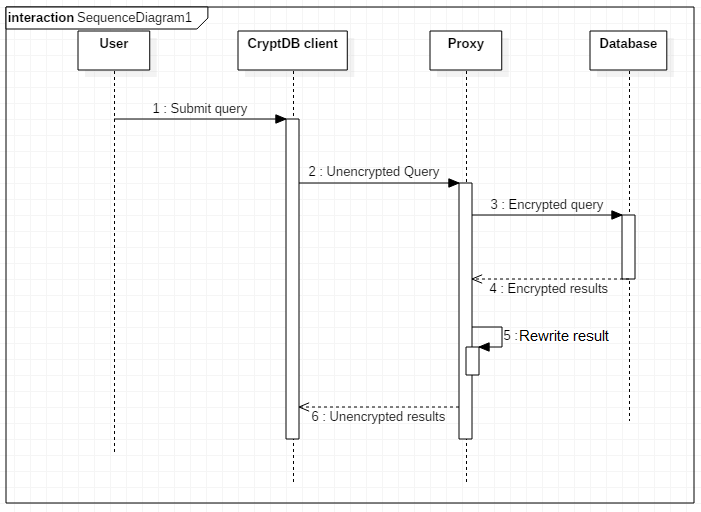
### #692 Create Basic Web App for CryptDB

#### Use Case Diagram



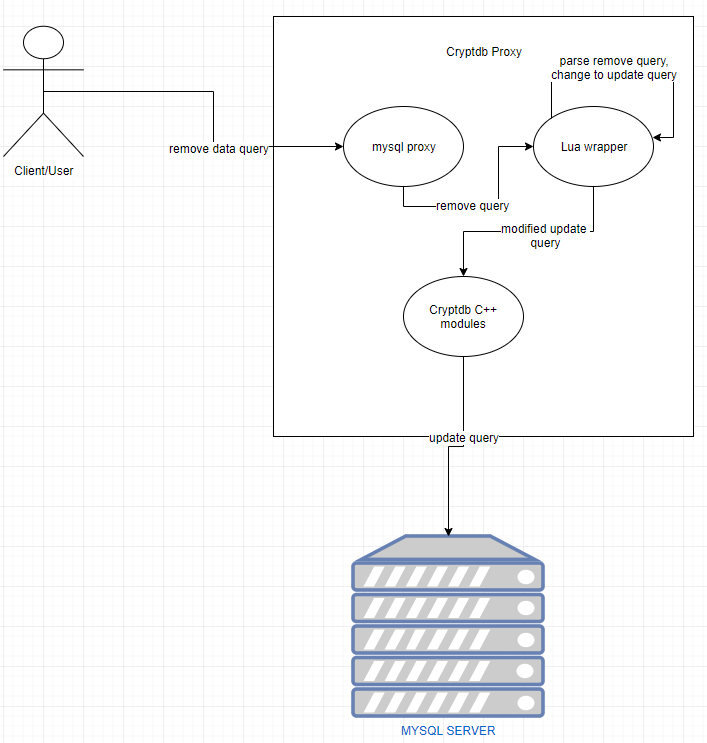
### #691 Omit Fake Attribute From Result Set

#### Sequence Diagram

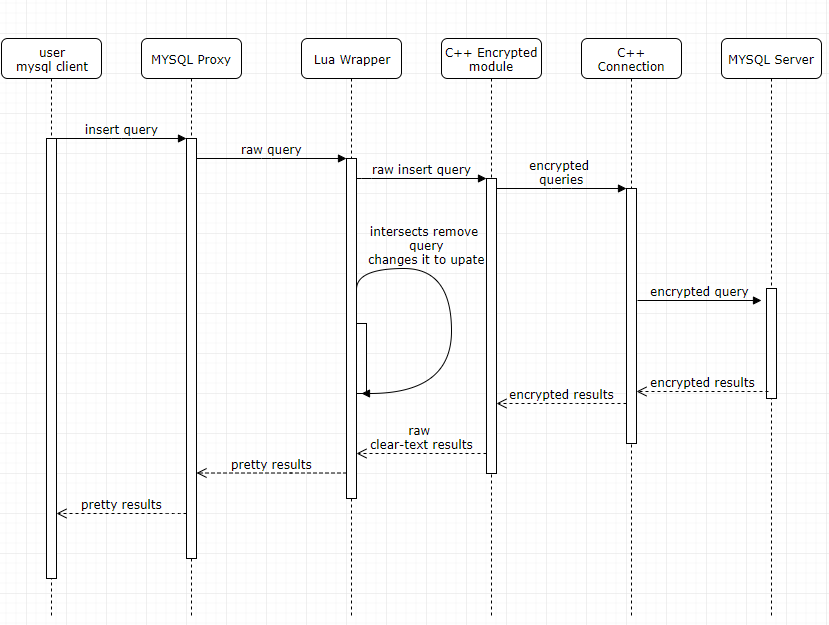


### #690 Parse Remove Query, and Modify to Update

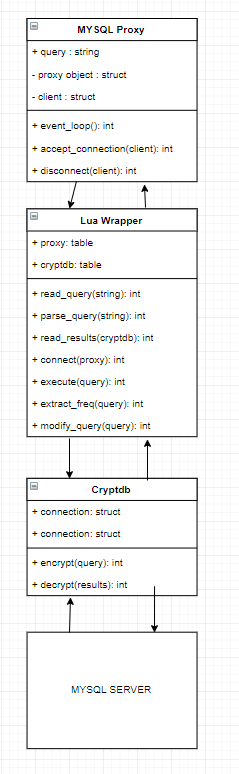
#### Use Case Diagram

****

#### Sequence Diagram

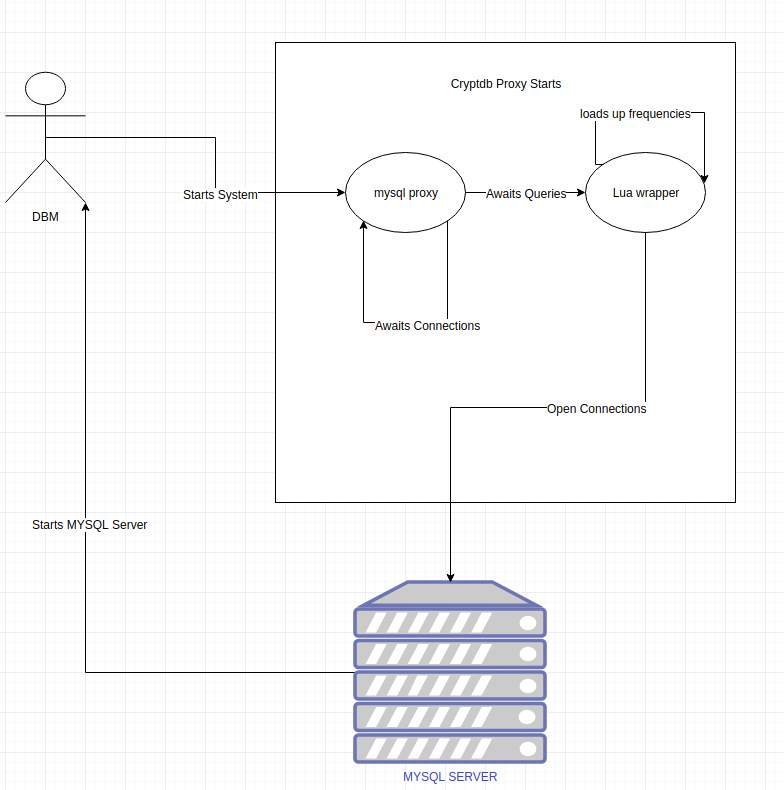
****

#### Class Diagram

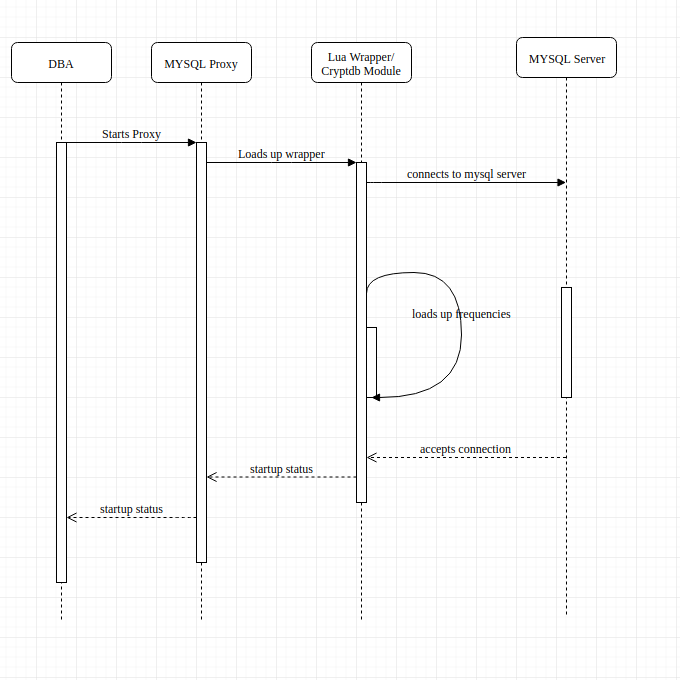
****

### #689 Store Frequencies in a File and Load them up at Startup

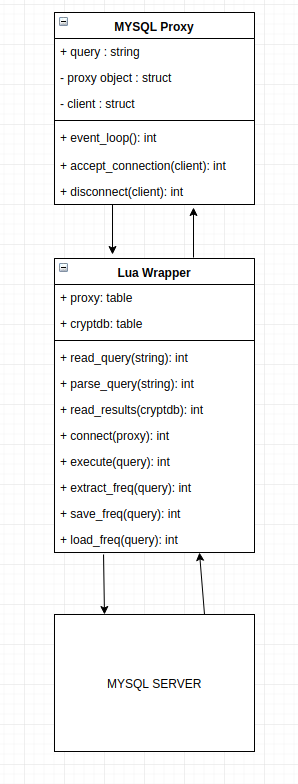
#### Use Case Diagram

****

#### Sequence Diagram

****

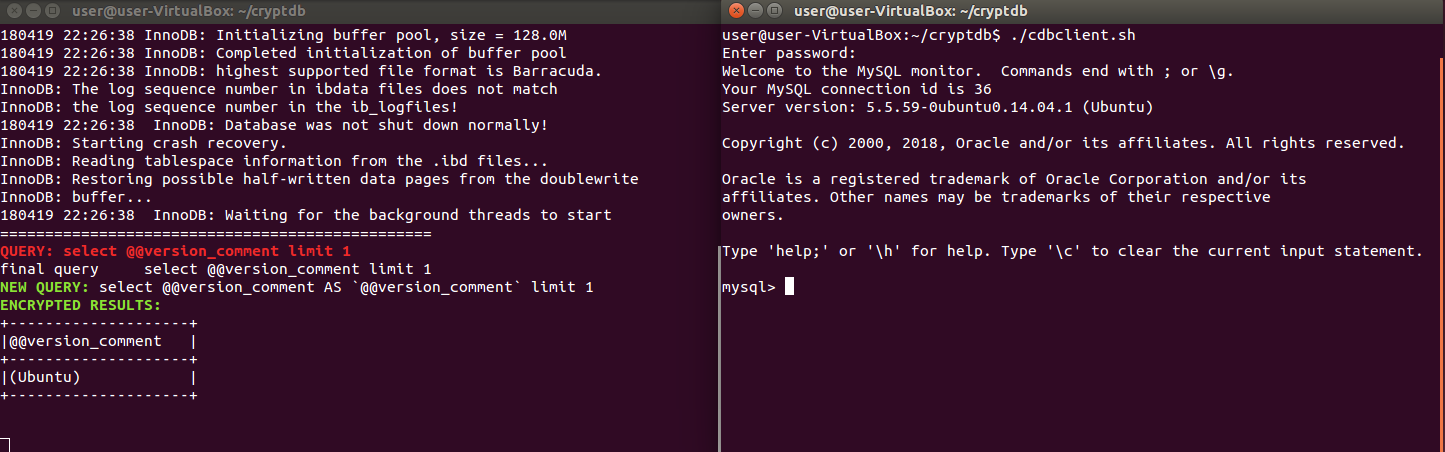
#### Class Diagram

****

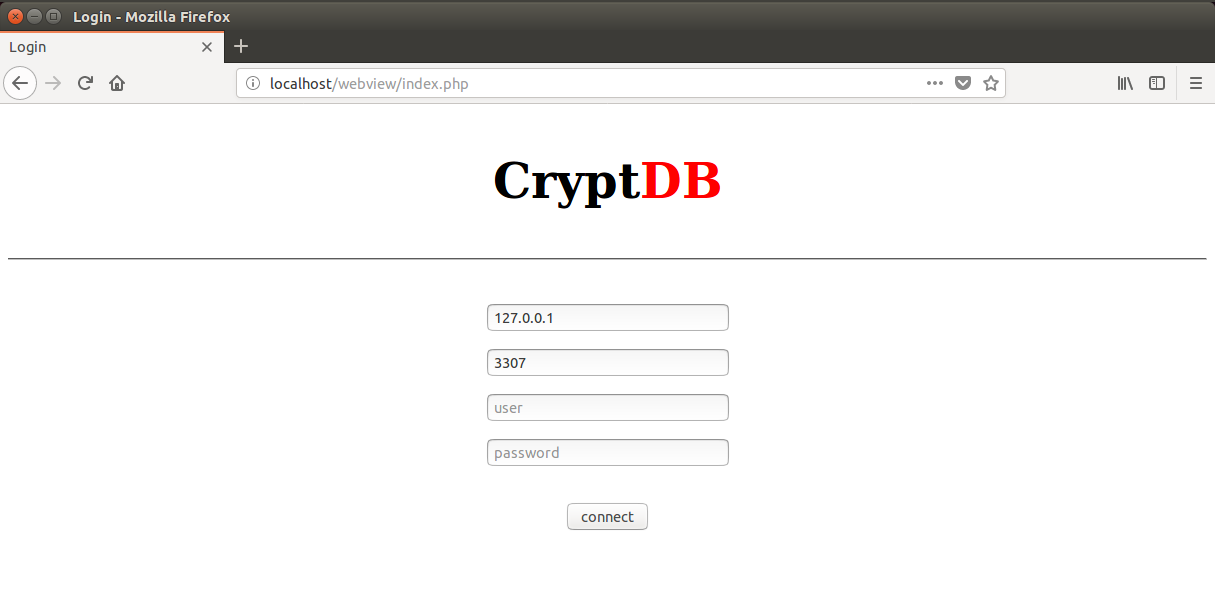
## Appendix B - User Interface Design

There are two user interfaces available for the CryptDB system:

**CLI:** We are able to connect to the CryptDB system through the terminal, same as MySQL, and communicate with the server through the command line.



**Web app:** A basic web application has been created for the CryptDB system, and allows the user to view, create, and drop database and tables. It is also able to view the values of a table, and the layers of encryption being used on encrypted columns.



## Appendix C - Sprint Review Reports

### Sprint Review Meeting Minutes (Sprint 1)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Run diff between practical CryptDB and the original CryptDB
* Generate data to insert into the database for testing
* Create frequency analysis script

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* None

### Sprint Review Meeting Minutes (Sprint 2)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Run diff between practical CryptDB and the original CryptDB
* Generate data to insert into the database for testing
* Create frequency analysis script

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* None

### 

### Sprint Review Meeting Minutes (Sprint 3)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Re-install CryptDB from the original #677
* Insert data intro CryptDB using annotations #678
* Debug web view #679
* Re-run frequency attack with the original CryptDB #680

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* None

### Sprint Review Meeting Minutes (Sprint 4)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Research areas to pivot the project - #681
* Research usefulness of onion web view - #684
* Research usefulness of sensitive annotations #682
* Research usefulness of fake data insertion as a countermeasure #683

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* Run debugger to see what is wrong with the annotations
* Keep working on onion web view

### Sprint Review Meeting Minutes (Sprint 5)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Read and parse query - #685
* Generate fake data to insert - #686
* Auto create fake/real data column and store values #682
* Filter out results that are fake #683

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* N/A

### Sprint Review Meeting Minutes (Sprint 6)

Attendees:

Joannier Pinales, Steven Caceres, Mireya Jurado, Mark Finlayson, Rob Cunningham

Start time: 10:30am

End time: 11:00am

After a show and tell presentation, the implementation of the following user stories were accepted by the product owners: All.

* Store frequencies in a file and encrypt the file- #689
* Intercept the remove query and modify to update fake -1 - #690
* Omit fake column from the result set - #691
* Create a basic web app to view the database -#692

The following ones were rejected and moved back to the product backlog to be assigned to a future sprint at a future Sprint Planning meeting.

* Discard the CryptDB C++ files to implement the insertion

## Appendix D - User Manuals, Installation/Maintenance Document, Shortcomings/Wishlist Document and other documents

### Installation guide and User Manual

##### How to get started using Docker:

1. Install Docker from <https://docs.docker.com/install/>
2. Build docker image (Open the Docker Quickstart Terminal if using OS X or Windows)

- *sudo docker build -t CryptDB:v1 .*

1. Run Docker, do one of these options
   1. Automatic

- *sudo ./run.sh*

* 1. Manual

- *docker run --name CryptDB\_v1 --volume $(pwd)/data:/opt/CryptDB/data/ -d -p 3306:3306 -p 3307:3307 -e MYSQL\_ROOT\_PASSWORD='letmein' CryptDB:1.0*

***-*** *docker exec -it CryptDB\_v1 bash*

These commands create a docker container bounding the ports 3306 and 3307 to the local machine where Docker is being ran. The password must be ‘*letmein’*, according to CryptDB formal documentation

##### How to get started using VM:

1. Downloading and installing required software.
   * Download and install VirtualBox from <https://www.virtualbox.org/wiki/Downloads>
   * Download a desktop image of Ubuntu14 from <http://releases.ubuntu.com/14.04/>
   * Open VirtualBox and create a new machine
     + Type: Linux
     + Version: Ubuntu
     + Hard disk: Create a virtual hard disk now
     + Allocation of 15 GB is recommended
2. Set up the machine using the Ubuntu 14 image
   * Select the new machine, and click start
   * Select the Ubuntu image, if not already selected, and click start
   * Once the machine boots, follow on-screen instructions to install Ubuntu
3. Updating the software
   * On first reboot, the system will prompt you to upgrade to Ubuntu16. Decline.
   * To update the system, there are two options:
     + Launch software updater
     + Launch terminal, and use the commands:

- sudo apt update

- sudo apt upgrade

1. Downloading and installing required software for CryptDB
   * Clone the main repo

- sudo apt install git  
- git clone <https://github.com/EncryptDB-Research/cryptdb-attacks>

* + In terminal, navigate to the CryptDB-attacks folder and run setup.sh

- cd CryptDB-attacks  
- sudo ./setup.sh

This will install Bison 2.7, ruby, git, apache2, php5, download our copy of CryptDB and install it.

1. (Optional) Install the webview:

- sudo cp -r ~/CryptDB-attacks/webview/ /var/www/html/

This will copy the webview directory to the apache root. To use the webview, launch the browser, and go to localhost/webview

##### 

##### Running CryptDB

Running the server and client

* Server

- bash proxy.sh

* Client

- bash client.sh

Each of these commands is required to be ran in a separate terminal. If the current computer does not have mysql client, then open a second terminal inside docker by running

- sudo docker exec -it CryptDB\_v1 bash

Then run the client command stated earlier. In a regular VM the client command can be run from in one of the VM's terminals. To run everything in one terminal then do

- bash proxy &  
 - bash client.sh

Two separate terminals is highly recommended to see CryptDB server output.

##### Inserting data with python (Optional)

1. Install python dependencies

- bash ./data/setup.sh

This will install the necessary python dependencies

1. Run the following commands to insert the data

- cd data  
 - python insert.py # regular insertion  
 - python insert\_sensitive.py # sensitive insertion

##### 

##### Frequency Analysis Guidelines

Overview:

To run frequencies analysis, two different insertion scripts were created, one insert data normally and the other one inserts data using CryptDB sensitive annotation schemas. Using pandas data frame the scripts load up around 5k records of names. Before inserting each data item a set of 4 different diseases is distributed throughout with the desired probability distribution for later comparison after the attack. After all the data is inserted then some queries are run to peel off the random layer from the column at test to leave it at the DET layer. The point of doing this is so that all the same values in the column get encrypted to the same ciphertext. Then using stats.py, the frequencies of each value in the column are extracted.

Requirements before doing frequency analysis:

* CryptDB installed and functioning properly
* Data Inserted

Step by Step:

Inside mysql client run

- use <<Database Name>>;

- select \* <<from table name>> where <<illness>>='cancer';

This will peel off the random layer from the column being tested and put leave at DET encryption. To confirm this behavior, run the command below.

- set @cryptdb='show';

This should show the column under test at the DET encryption. Then, in a regular terminal run stats.py file in the data folder. From here on there's a little bit of guessing, since the table and column names are encrypted.

While stats.py is running:

1. Select database being analyzed
2. Pick the first encrypted table name.
3. From the list of encrypted columns, pick one that contains oEQ in the name (which stands for DET encryption). Then the script will give a count of distinct values in the column,
4. If the count is equal the total count then the layer still with random encryption, so move on to a different column name that contains oEQ.
5. If ran out of columns with oEQ then go back to the table selection and select a different table.
6. Repeat step 4 until finding a column where the count of unique values is less than the total count.
7. Then the script will prompt to save the frequencies, select yes and check the file to see the frequencies obtained from the column.

### Wishlist Document

1. Create an annotation for the user to specify which columns should be adjusted. Right now the columns to be checked are hard coded.
2. Implement ***remove*** and ***update*** queries to not affect the histogram. A way to handle the remove query is to change it to an update query and change the fake to true for the respective item being remove. However this could mean that any user data is not truly deleted from the database which can lead to privacy issues.
3. Look into random insertion instead of keeping the histogram flat at all times. As of now whenever a new value is inserted into the column its frequency gets adjusted to the maximum frequency. This could be changed to randomly pick a number between 0 and the maximum and insert that quantity. By doing this it will not create flat histogram but it will create a randomized histogram which should also prevent frequency analysis against the column.
4. Store the frequencies in the database instead of file for maximum security. As of now the frequency are stored in a file with clear text. Instead they should be encrypted and stored in the database.
5. Looking into deleting unnecessary fake data periodically without keeping track of fake values count. A possible algorithm is to do a select all query then save the id of the ones that are fake for each column value. Then calculate the min number of fake values that can be deleted for each value without affecting the flat histogram. This number is calculated by getting the total count of a column subtracted by the value with the max number of non-fake values.
6. Improve the insert query parser to catch different variations of the insert query. As of now the insert parser only works with ***insert tablename(col1, col2, ...) values (val1, val2)***. Improve the parser to catch other variations of this query. Also if the query fails, the count still updates, which should not happen.
7. Improve the fake data filtering to handle queries which don’t specifically call the fake column, or returns the fake column in the result set. The current version of this implementation assumes that the result set will contain the fake column. Possible fix is to rewrite the query to include fake in the result set for every query, but may prove to be difficult with nested queries.
8. Web app: Has the ability to insert values, but messes up the frequency tracker since it does not include the attribute names in the query (refer to item 6). As of now, the submit button for changing the encryption layer is commented out, but is able to send the query to adjust the encryption. CryptDB, however, receives the query and begins adjusting, but does not fully commit the change.

# 

# References

1. **Raluca Ada Popa, Catherine Redfield, Stephen Tu, Hari Balakrishnan, Frans Kaashoek, Sam Madden, Nickolai Zeldovich, Aaron Burrow. (2011, October).** [**https://css.csail.mit.edu/cryptdb/**](https://css.csail.mit.edu/cryptdb/)
2. **Cryptdb-org. (2013, June 18). Cryptdb-wiki.** [**https://github.com/cryptdb-org/cryptdb-wiki/wiki**](https://github.com/cryptdb-org/cryptdb-wiki/wiki)
3. **Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. (2011, October). Protecting Confidentiality with Encrypted Query Processing.** [**http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf**](http://people.csail.mit.edu/nickolai/papers/raluca-cryptdb.pdf)
4. **Raluca Ada Popa, Nickolai Zeldovich, and Hari Balakrishnan. (2015). Guidelines for Using the CryptDB System Securely.** [**https://eprint.iacr.org/2015/979**](https://eprint.iacr.org/2015/979)
5. **Bipinu. (2015, April). Install bison 2.7 on Ubuntu 14.04.** [**https://gist.github.com/bipinu/520af22ff90994962df7**](https://gist.github.com/bipinu/520af22ff90994962df7)
6. **Naveed Kamara, Inference Attacks on Property-Preserving Encrypted Databases** [**https://cs.brown.edu/~seny/pubs/edb.pdf**](https://cs.brown.edu/~seny/pubs/edb.pdf)
7. [**https://en.wikipedia.org/wiki/Inference\_attack**](https://en.wikipedia.org/wiki/Inference_attack)
8. **M. Cooney. IBM touts encryption innovation; new technology performs calculations on encrypted data without decrypting it. Computer World, June 2009.**
9. **P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the 18th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Prague, Czech Republic, May 1999**
10. **D. X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In Proceedings of the 21st IEEE Symposium on Security and Privacy, Oakland, CA, May 2000**