



**Partial query processing over encrypted data with Object
Relational Mapping.**

معالجة جزئية للاستعلام على البيانات المشفرة مع ORM

Student Name: Iyad Naim

Supervisor

Prof. Ahmad Kayed

**A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master, in Computer Science**

Department of Computer Science

Faculty of Information Technology

Middle East University

January 2016

Dedication

I, Iyad Naim, authorize the Middle East University to provide hard copies or electronic copies of my thesis to libraries, institutions or people when asked.

Name: Iyad Naim

Date: January 30th, 2016

Signature:

A handwritten signature in blue ink, appearing to read 'Iyad Naim', with a long horizontal flourish extending to the right.

Middle East University

Examination Committee Decision

This is to certify that the thesis entitled “Partial Query Processing over Encrypted Data with Object Relational Mapping” was successfully defended and approved in 2016.

Examination Committee Members

(Supervisor)

Professor Ahmad Kayed
*Dean Faculty of IT
Middle East University*

Signature



(Chairman of Examination Committee)

Dr. Sadeq AlHamouz
*Head of Computer Information System
Middle East University*



(External Committee Members)

Dr. Shadi Aljawarneh
*Software Engineering
Jordan University of Science and Technology*



Acknowledgements

MSc is a great opportunity and experience. It is tough, but I am truly happy and honored to complete it. I could not have done it without all those people who helped me along the way. First of all, I would like to thank my supervisor, Dr. Ahmad Kayed, who has guided me and advised me throughout my research period.

Also I would like to express my appreciation to Dr. Sadeq Alhamouz & Dr. Shadi Jawarneh for their suggestions and comments on improving my work.

Finally, I would like to thank my great family for their support.

DEDICATION

To all those who provided me with love, hope and help...

To all those who support my way to knowledge...

To my beautiful family, beloved mother, brothers and sisters.

& To all those who sacrificed their rights to achieve this work...

Table of Contents

AUTHORIZATION STATEMENT.....	II
EXAMINATION COMMITTEE DECISION	III
ACKNOWLEDGMENT	IV
DEDICATION.....	V
TABLE OF CONTENT.....	VII
LIST OF FIGURES	IX
LIST OF Abbreviations.....	X
ABSTRACT.....	XI
الملخص باللغة العربية	XIII

Chapter One (Introduction)

1.1. Introduction	2
1.2. Research Problems.....	3
1.3. Research Objectives	4
1.4. Motivation.....	4
1.5. Research Methodology	5
1.6. Thesis Layout.....	6

Chapter Two (Background and Literature Review)

2.1. Preface	8
2.2. Background	8
2.2.1. Cloud Computing	8
2.2.2. Security In Cloud Computing.....	10
2.2.3. Relational Database	10
2.2.4. Object-Oriented Design	11
2.3. Related Work	14

Chapter Three (The Proposed Framework)

3.1.	Preface	20
3.2.	CryptDB	20
3.3.	Layered Architecture.....	23
3.4.	Object Relational Mapping Overview	25
3.4.1.	What do we mean by persistence?.....	25
3.4.2.	ORM Mapping example	27
3.4.3.	Mismatching between the tables and the classes in the object oriented	27
3.5.	Choosing our ORM Library	30
3.6.	Implementation	31

Chapter Four (The Discussion of Results)

4.1.	Preface	40
4.2.	Comparing The Result With The Previous Work (CryptDB)	40
4.3.	Contribution.....	46
4.4.	Work Limitations.....	47

Chapter Five (Conclusions and Future Work)

5.1.	Preface	49
5.2.	Conclusions	49
5.3.	Recommendations For Future Work	50

List of Figures:

Figure 1 Passive DB Server Attacks	21
Figure 2 Any Attacks on All Servers	21
Figure 3 Onions In CryptDB	22
Figure 4 Three Columns Example In CryptDB	23
Figure 5 Application Architecture Layers	24
Figure 6 Application Architecture Layers With ORM	25
Figure 7 ORM Mapping Example	27
Figure 8 Granularity Example	29
Figure 9 Creating Table In CryptDB Project.....	33
Figure 10 Trusted Server In CryptDB Project 1	35
Figure 11 Untrusted Server In CryptDB Project 1	35
Figure 12 Trusted Server In CryptDB Project 2	36
Figure 13 Untrusted Server In CryptDB Project 2	37
Figure 14 Inserting Data In CryptDB Project	37
Figure 15 Creating Table In CryptDB Project.....	38
Figure 16 Deleting Table In CryptDB Project.....	38

List of Abbreviations

ORM	Object-relational mapping
SAAS	Software As A Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
SQL	Structured Query Language
DDL	Data Definition Language
POC	Proof Of Concept
ADO	ADO ActiveX Data Objects
DAL	Data Access Layer
DLL	Dynamic-link library
MRSE	Multi-keyword Ranked Search Over Encrypted data

Partial query processing over encrypted data with Object Relational Mapping.

Student Name: Iyad Naim

Supervisor
Prof. Ahmad Kayed

Abstract

With the great power of the cloud computing, including its scalability and accessibility anywhere, anytime, from any device; resource outsourcing, to the cloud computing becomes more popular. However, most organizations that hold sensitive information such as government offices still have their own data center which is run by an in-house IT department. Fear of security incidents is the main concern prevent them from using the cloud computing.

Obvious solution to secure our data, is to encrypt it, but the issue here, after the encryption we will not be able to perform our usual operations such as searching and editing. CryptDB performs operations over the encrypted data. On the other hand, one of the popular concepts in the software engineering called Object Relational Mapping, (ORM). It will automate the process of creating functions to access and manage your relational database.

The main idea of this thesis is to restructure CryptDB using the object relational mapper framework. This thesis merged both ORM and CryptDB together. This merging will deploy the benefits of ORM during the development stage for any future use of CryptDB. The proposed

development framework will handle both encryption and decryption of the data, with the ability of performing queries over the encrypted data. These have been done by tracing the CryptDB basic operations (create, read, update & delete). Then generates the required SQL (e.g. select statement in SQL) using ORM concept. After that, intercept and rewrite the generated scripts to be compatible with the CryptDB security concepts.

Keywords: Query Processing Over Encrypted Data, Encryption & Decryption with Object Relational Mapping.

معالجة جزئية للاستعلام على البيانات المشفرة مع ORM

إعداد

إياد نعيم

إشراف

الأستاذ الدكتور أحمد كايد

الملخص

في الوقت الحاضر ومع القوة الهائلة للحوسبة السحابية بما في ذلك إمكانية التوسع وسهولة الوصول إليها من أي مكان، في أي وقت، ومن أي جهاز. أصبحت الحوسبة السحابية أكثر شعبية، ولكن في الواقع معظم المنظمات التي لديها معلومات حساسة مثل المكاتب الحكومية والمؤسسات المالية، لا تزال لديها مراكز البيانات الخاصة بها، والتي يتم تشغيلها من قبل قسم تكنولوجيا المعلومات في المؤسسة. الخوف من الحوادث الأمنية هو أكبر سبب يحول دون اعتماد الحوسبة السحابية في تلك المؤسسات.

الحل الواضح لتأمين بياناتنا، هو تشفيرها، ولكن المسألة هنا، بعد تشفير البيانات لن تكون قادرين على أداء العمليات المعتادة عليها مثل البحث والتحرير، مشروع CryptDB. يسمح لنا بنفيذ عمليات على البيانات المشفرة. من ناحية أخرى، واحدة من المفاهيم الشائعة في هندسة البرمجيات يدعى (Object Relational Mapping). الذي يعمل على إنشاء دالات بطريقة اتوماتيكية، هذه الدالات تسمح بالوصول وإدارة قواعد البيانات العلائقية.

الفكرة الرئيسية في هذه الأطروحة هي إعادة هيكلة مشروع CryptDB باستخدام ORM، ما قمنا به هو دمج كلاً من ORM و CryptDB معاً، الأمر الذي يسمح لنا بالاستفادة من قدرات ORM خلال مرحلة تطوير التطبيقات، وفي نفس الوقت، النتيجة التي حصلنا عليها تتيح لنا كلاً من التشفير وفك التشفير للبيانات، مع القدرة على أداء استفسارات على البيانات المشفرة. وقد تمت هذه عن طريق تتبع العمليات الأساسية في CryptDB (خلق، وقراءة، وتحديث وحذف). ثم إنشاء جمل

SQL المطلوبة (على سبيل المثال جملة Select في SQL) باستخدام مفاهيم ORM. بعد ذلك، اعتراض وإعادة كتابة النصوص الناتجة لتكون متوافقة مع مفاهيم الأمن الخاصة ب CryptDB.

كلمات البحث: معالجة الاستعلام على البيانات المشفرة مع ORM ، تشفير و فك التشفير مع ORM.

CHAPTER ONE

Introduction

Introduction

Cloud computing is a general term for anything that involves delivering hosted services over the Internet Sathyavani & Senthilkumar (2013). Those services could be one of the following: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Services (SaaS). Those services are used in Public Cloud, Private Cloud, Hybrid Cloud and Community Cloud Rani & Ranjan, (2014).

In fact, the associated security risks for the cloud computing still prevent many potential users from deploying it Evdokimov & Günther (2007), one possible solution is to encrypt the data before outsource it, the question here, can the data owner benefit from his data after encrypt it. This forced to think, how to encrypt our data in such a way that allow us to perform some operations like search over the encrypt data.

One of the previous systems that provide practical and provable confidentiality in the face of attacks for applications backed by SQL databases called CryptDB. It works by executing SQL queries over encrypted data using a collection of efficient SQL-aware encryption schemes. As a result, hackers and curious database administrators never obtains access to decrypted data, and even if all servers are compromised Popa, Redfield, Zeldovich, & Balakrishnan (2011).

On the other hand, the object-oriented design is mainstream approach that is used for application development; usually those applications have to store their data. If talk

about enterprise applications, we can say that it is usual choice to use relational databases as their default data storage. As there is a gap between object-oriented design and relational database. Object relational mapping is in place, especially when we deal with large information systems that have to be open and easy to maintain Jaroslav (2006). Entity Framework & NHibernate used in .NET as an ORM framework while Hibernate used with Java.

1.1 Research Problems

The Object Relational Mapping technique resolves the mismatching between the tables in the relational database and the classes in the object oriented design. CryptDB project allows us to perform operations over the encrypted data. This research emphasized on, restructuring CryptDB project using the Object Relational Mapping technique. The challenge is, how to create a new development framework, which will apply CryptDB concepts, including encryption and decryption, simultaneously this development framework must allow the developers benefit from the original Object Relational Mapper capabilities during the development phase.

The problem will be accomplished by answering the following questions:

- How to interrupt real time SQL queries that generated from the object relational mapper.
- How to rewrite database schema, to be incomprehensible schema.

- How to rewrite (encrypt & decrypt) the interrupted SQL including user input values, to be compatible with the encrypted database.

1.2 Research Objective

The main propose of this thesis is to create a new development framework, which will allow the developers to connect the Object Relational Mapper with encrypted database, this will encourage the developers to build secure applications with the benefits of the Object Relational Mapper capabilities, such as enhancement of the developer productivity, especially in CRUD (Create, Read, Update, and Delete) applications, allowing developers to focus on the business logic of the application rather than repetitive CRUD logic.

1.3 Motivation

As far as known, we have presented the first ORM developments framework which working with encrypted database. To encourage programmers to develop new secure applications in very fast and easy way, without taking care of both encryption and decryption process. This will increase the possibility of chosen the cloud as their default environment, which will allow them to benefit from the ORM capabilities.

Research Methodology

The main idea in this research is to create new development framework which will merged both Object Relational Mapper library and CryptDB project together, for the main time we are going to apply CryptDB concept for the CRUD operations. Research methodology will involve the following the following steps:

1. Installing and building CryptDB project.
2. Choose Nhibernate as our experimental ORM, then build its source code.
3. Tracing Nhibernate algorithms, then adding interceptor to get real time queries from the object relational mapper.
4. After interrupting the generated SQL, Rewrite database schema, to be incomprehensible schema be applying CryptDB database concepts.
5. Encrypt and decrypt the interrupted SQL including user input values, to be compatible with the encrypted database.
6. Test and evaluate the resulted development framework.
7. Compare database result with the original CryptDB project database result.

1.4 Thesis Layout

The layout of the thesis corresponds to the structure of the work, as undertaken throughout the study. Following this introductory chapter, the thesis includes five more chapters, references and appendices. A brief content description for the five chapters is presented below:

Chapter 2 provides the summary of the literature review and related works that are related to the MSc project to fully understand the main aspects. It details and analyses to the project elements and identifies all of the factors for each element and their assessment methods. In addition, this chapter also highlights the main shortcoming found from both this stage and the literature review stage.

Chapter 3 provides the design of the proposed model and includes the main performance factors with their assessment methods. Furthermore, chapter 3 includes overview for CryptDB project, after that we are going to discuss software engineering aspects like layered architecture and Object Relational Mapping.

Chapter 4 presents the method and the main outcomes of the test. Including analysis and compare the result, between the original CryptDB project and the proposed development framework using NHibernate, which will enhance the developer productivity especially in CRUD applications.

Chapter 5 concludes the project and suggests future work in order to improve the final setup and lead to a better implementation of the system.

CHAPTER TWO

Background and Literature Review

2.1 Introduction:

This chapter provides a background and literature review on the main concepts covered by this research. It is divided into three sections. Section 2.2 discusses the necessary background information that is needed to better understand the topics embedded in the thesis such as cloud computing since the thesis idea is to outsource the database to the untrusted cloud, then some database and software engineering concepts which are related to ORM. The most related studies in the field of cloud computing security and Object-relational mapping are discussed in section 2.3.

2.2 Background

2.2.1 CLOUD COMPUTING

Usually, for software to run on a computer, an individual copy of the software had to be installed on the computer, either from a disk or, more recently, after being downloaded from the Internet. Nowadays the concept of cloud computing has changed this. For example, if you access your e-mail via your web browser, you are using a form of cloud computing. If you use Google Drive's applications, you are using cloud computing, if you are surfing social networking sites, you are using cloud computing David (2014).

Advantages of Cloud Computing:

- No software to install or upgrades to maintain.
- Available from any computer that has access to the Internet.

- Can scale to a large number of users easily.
- New applications can be up and running very quickly.
- Services can be leased for a limited time on an as-needed basis.
- Your information is not lost if your hard disk crashes or your laptop is stolen.
- You are not limited by the available memory or disk space on your computer.

David (2014).

Types of cloud computing

The following are the types of cloud computing available:

- Public cloud: this is when cloud providers make various resources available to the public to use and register. Cloud services provided are sometimes free or not Sabahi (2011).
- Private cloud: this is when cloud services are made available for the internal use of specific organizations and not made available for public use Sabahi (2011).
- Hybrid cloud: in hybrid cloud, the cloud provider's service is in two parts; one part is the private part that can only be accessed by authorized users and the other part which is the public part can be accessed by members of the public Sabahi (2011).

2.2.2 SECURITY IN CLOUD COMPUTING

Cloud Computing security is the major concern to be addressed nowadays. If security measures are not provided properly for data operations and transmissions, then the data are at high risk Velumadhava & Selvamanib (2015), lots of organizations are holding back from adopting the technology due to security issues and concerns Sheriff (2014). This research will provide data confidentiality for both:

- Hackers even if all servers are compromised.
- Curious database administrators.

2.2.3 RELATIONAL DATABASE

One of the most important applications of computers is store and manage information. The manner in which information is organized can have a profound effect on how easy it is to access and manage Jeffrey & Alfred (1992). In the relational database Michael (2013):

- A table is the main structure. It is composed of fields and records, the order of which is completely unimportant. It always represents a single, specific subject, which can be an object or an event.
- A field (attribute) is the smallest structure in a relational database. It represents a characteristic of the subject of the table. A field may be multipart, multi-valued or the result of a calculation (concatenated).

- A record (tuple) is a structure within a table that represents a unique instance of the subject of the table.
- A View is a virtual table that is composed of the fields of one or more tables. It draws its data from the tables on which it is based. They are commonly implemented as saved queries.

Types of Table Relationships In the relational database Microsoft (On-Line).

A relationship works by matching data in the key columns usually columns with the same name in both tables. In most cases, the relationship matches the primary key from one table, which provides a unique identifier for each row, with an entry in the foreign key in the other table.

There are three types of relationships between tables:

- One-to-Many Relationships

A one-to-many relationship is the most common type of relationship. In this type of relationship, a row in table A can have many matching rows in table B, but a row in table B can have only one matching row in table A.

- Many-to-Many Relationships

In a many-to-many relationship, a row in table A can have many matching rows in table B, and vice versa. You create such a relationship by defining a third table, called a junction table, whose primary key consists of the foreign keys from both table A and table B.

- One-to-One Relationships

In a one-to-one relationship, a row in table A can have no more than one matching row in table B, and vice versa. A one-to-one relationship is created if both of the related columns are primary keys or have unique constraints.

2.2.4 OBJECT-ORIENTED DESIGN

Object-oriented design is a programming paradigm that began in the late 60's, as software programs became more and more complex. The idea behind the approach was to build software systems by modeling them based on the real-world objects that they were trying to represent, and it is widely accepted that object oriented development requires a different way of thinking than traditional structured development. The main advantage of object oriented design is its modularity and reusability Manoj, Ravinder & Manish (2014).

Object-oriented concepts (principles) William & David (1998):

An object is a thing about which data are stored and manipulated. It might be a physical thing such as a person, a customer, a book, or an item in inventory. It might be an abstract thing such as a model, a concept, or a process. Unlike many technical terms, the word object means what people intuitively think it means.

- **Encapsulation:**

Both data and methods are associated with an object. For example, an item in inventory might be described by listing such attributes as its product code, a brief description, its selling price, and so on. A method is a process that accesses an object. For example, associated with a given product in inventory are methods for placing it in inventory, changing one of more of its attributes, removing it from inventory, and so on. Methods to define how the object's data are manipulated. In

an object-oriented program, an object's data and methods are bundled so that the only way to access the data is through the object's own methods. Hiding implementation details in this way are called encapsulation. The only way other objects can obtain a given object's data is through one of that object's own methods William & David (1998).

- **Inheritance:**

A Mazda Miata can be described as a small, sporty, two-seat automobile. Because it is a type (subclass) of automobile, all the attributes the Miata shares with other automobiles (four wheels, an engine, a cooling system, methods for propulsion, steering, and stopping) can be assumed. Moving down another level, a specific Mazda Miata (an object) can be described in terms of the attributes that make it unique (red, convertible top, serial number), and the attributes it shares with other Miatas (small, two seats, sporty) can be assumed. In effect, each subclass borrows (or inherits) attributes and methods from its superclass. This concept is called inheritance William & David (1998).

- **Objects and object types:**

To avoid being swamped by the sheer number of objects, similar objects are grouped to form classes or object types. Classifying or grouping objects makes it easier to track them. An individual object is a single instance (or occurrence) of an object class (or object type). For example, a given computer (the object itself) has a unique serial number. That particular computer is but one instance of a given model. Moving up the classification hierarchy, a store might distinguish between tower,

desktop, laptop, and hand-held computers. Finally, computers, printers, boards, software, supplies, books, and services clearly represent different categories.

- **Polymorphism:**

A given operation or method is considered polymorphic if it produces similar results in different objects or at different levels. For example, a customer sale, a customer return, the arrival of a shipment, and the completion of a physical inventory are all events that can change the value of the inventory stock-on-hand for a given object type. The general structure of the inventory update method might be inherited from the highest-level class, and then customized for each of these special cases William & David (1998).

2.3 Related Work

Many researchers presented many concepts and tools trying to improve data privacy, especially in the cloud computing. A literature review presents to our problem as listed below:

CryptDB provides the first practical DBMS to process multiple queries; this DBMS can be used to hide the database from system administrators. While allowing them to perform their tasks on the encrypted database. However, CryptDB not compatible neither with stored procedure nor Object Relational Mapper. Although; they are very popular in the practical life. Our idea is to link between CryptDB project and ORM, to benefit from the object relational mapping capabilities Popa, Redfield, Zeldovich, & Balakrishnan(2011).

“MONOMI” is a system for securely executing analytical workloads over sensitive data on an untrusted database server; it encrypts the entire database and run queries over the encrypted data, the MONOMI use split client/server query execution, which can execute arbitrarily complex queries over encrypted data Tu, Kaashoek, Madden, & Zeldovich(2013).

Craig Gentry published a paper showing - for the first time - how to construct a fully-homomorphic encryption (FHE) scheme, solving a central open problem in cryptography. Such a scheme allows one to compute arbitrary functions over encrypted data without the decryption key Gentry(2009).

“New Security Models and Provably-Secure Schemes” propose methods for exact searches that do not require scanning the entire database and could be used to process certain restricted SQL queries Amanatidis, Boldyreva, & O’Neill (2007).

“An Ideal-Security Protocol for Order-Preserving Encoding” is an encryption scheme where the sort order of cipher texts matches the sort order of the corresponding plaintexts. To get effect query processing involving order over encrypted data Popa, Zeldovich, & Frank (2013).

Another work Instead of processing encrypted data directly, an alternative is to use an encrypted index, which allows the client to traverse the index and to locate the data of interest in a small number of rounds of retrieval and decryption, Although this work provides confidentiality for data residing in storage, they do not provide data confidentiality under dynamic query access patterns Univ(2007).

Several studies discussed mechanisms for processing transactional queries over encrypted data. But, a Few studies discussed how a data warehouse (DW) hosted in a cloud should be encrypted to enable analytical query processing. " Processing OLAP Queries over an Encrypted Data Warehouse Stored in the Cloud" present a novel method for encrypting a DW and show performance results of this DW implementation. Moreover, an OLAP system based on the proposed encryption method was developed and performance tests were conducted to validate our system in terms of query processing performance Lopes, Times, Matwin, Ciferri & Ciferri (2014).

Ning, Walmart, Cong, Ming & Kui define and solve the challenging problem of privacy-preserving multi-keyword ranked search over encrypted data in cloud computing (MRSE). They establish a set of strict privacy requirements for such a secure cloud data utilization system. Among various multi-keyword semantics. Related works on searchable encryption, focus on single keyword search or Boolean keyword search, and rarely sort the search results Cao, Labs, View, Wang, Li & Ren (2014).

Another paper talking about multi-keyword Search over encrypted data called "Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud" its focus on fuzzy search which mean finding result that match a pattern approximately. They propose a novel multi-keyword fuzzy search scheme by exploiting the locality-sensitive hashing technique. They proposed scheme achieves fuzzy matching through algorithmic design rather than expanding the index file. It also eliminates the need of a predefined

dictionary and effectively supports multiple keyword fuzzy search without increasing the index or search complexity Wang, Lou & Hou (2014).

Zhirong, Jiwu & Wei present for the first time, we investigate the challenging problem of preferred keyword search over encrypted data (PSED). They first establish a set of privacy requirements and utilize the appearance frequency of each keyword to serve as its “weight” We further represent keyword weights of each file by using vectors, convert the preference polynomial into the vector form, and securely calculate their inner products to quantitatively characterize the relevance measure between data files and a query Shen, Shu, & Xue (2013).

“Mylar” is a platform for building web applications on top of encrypted data, which protects data confidentiality against attackers with full access to servers. Mylar stores sensitive data encrypted on the server, and decrypts that data only in users’ browsers. Mylar addresses three challenges in making this approach work. Mylar allows the server to a perform keyword search over encrypted documents, even if the documents are encrypted with different keys. Recently Boston’s Newton-Wellesley hospital decided to use Mylar design to secure there data Popa, Stark, Helfer, Valdez, Zeldovich, Kaashoek, & Balakrishnan (2014).

As we sow there are many researchers discussing this topic, and we have built our works based on CryptDB project Popa, Redfield, Zeldovich, & Balakrishnan (2011). CryptDB project allows the developers to perform operations (e.g. search) over encrypted data. But the difference, CryptDB works in a traditional way, which the developer must

write the required SQL manually by his by hand, while the proposed work applies the same CryptDB concepts for the basic (create, read, update & delete) operations simultaneously the developer will benefit from the ORM capabilities which will automate the process of writing the required SQL. The idea behind "ZODB" Veitch, A. (2002) project was very close to the proposed work since both "ZODB" and the proposed work allow the developers to focus on business logic application, rather than writing a lot of database code, and there is no separate language for database operations. But the difference "ZODB" works with object oriented database, while the proposed framework works with relational databases.

CHAPTER THREE

The Proposed Framework

3.1 Introduction

This chapter presents the proposed framework which restructures both NHibernate as our experimental object-relational mapper and CryptDB Project (which already done by MIT - Massachusetts Institute of Technology). The aim of the proposed framework is to develop secure applications with the benefits of the ORM (Object Relational Mapping) capabilities. Including enhancement of the developer productivity, especially in CRUD.

3.2 CryptDB

One of the previous projects that already solved processing over the encrypted data called CryptDB. CryptDB provides practical DBMS to process multiple queries; this DBMS can be used to hide the database from system administrators. While allowing them to perform their task, or to outsource database to untrusted cloud Popa, Redfield, Zeldovich, & Balakrishnan (2011).

CryptDB addresses two threats. The first threat is a curious database administrator (DBA) who tries to get private data by snooping on the DBMS server; here, CryptDB prevents the DBA from knowing the private data. The current proposed framework focus on restructuring this threat to be compatible with the object relational mapper.

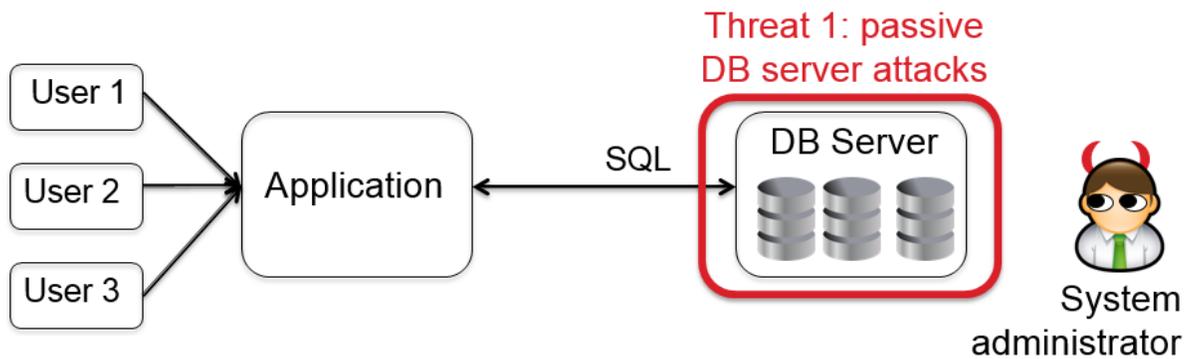


Figure 1 passive DB server attacks Popa, Redfield, Zeldovich, & Balakrishnan (2011).

The second threat is a hacker that gains complete control of application and DBMS servers. In this case, CryptDB cannot provide any guarantees for users that are logged into the application during an attack, but can still ensure the confidentiality of the logged-out users' data Popa, Redfield, Zeldovich, & Balakrishnan (2011)

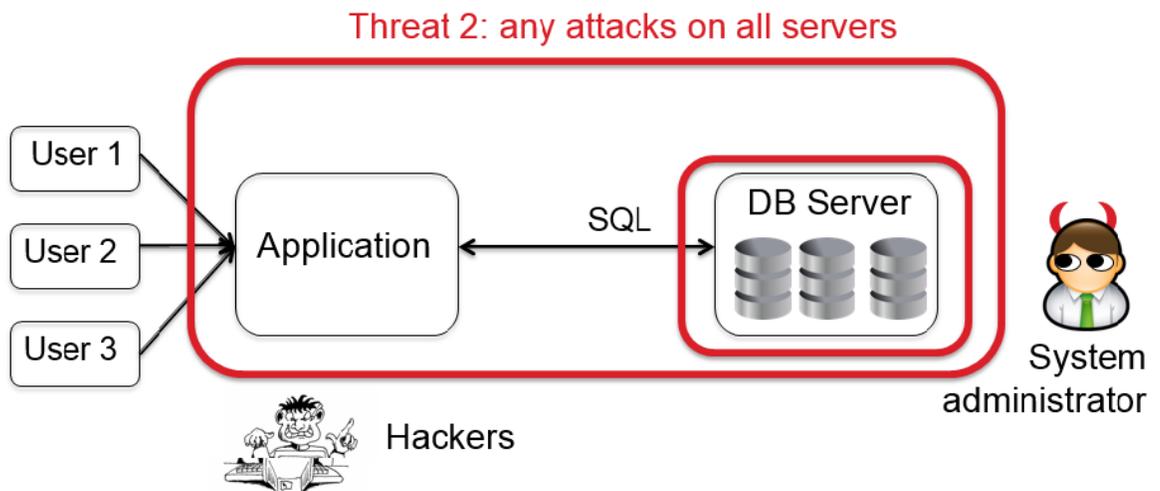


Figure 2 any attacks on all servers Popa, Redfield, Zeldovich, & Balakrishnan (2011).

CryptDB stack encryption scheme into onions. Onions are a novel way to compactly store multiple ciphertexts within each other in the database and avoid expensive re-encryptions Popa, Redfield, Zeldovich, & Balakrishnan (2011).

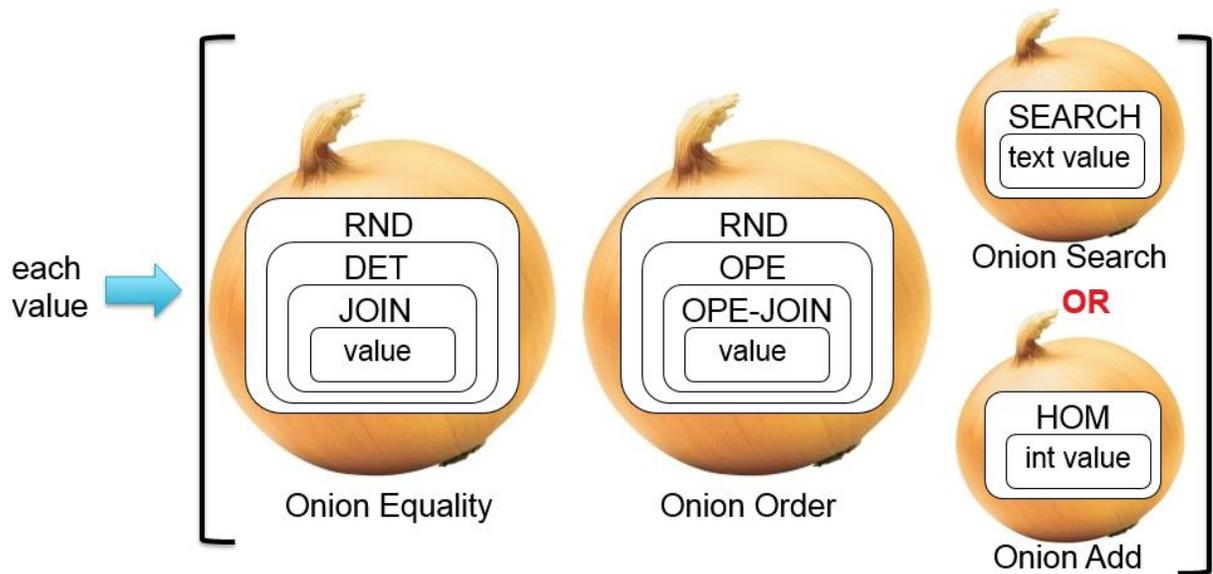


Figure 3 Onions in CryptDB Popa, Redfield, Zeldovich, & Balakrishnan (2011).

Each column become three columns each columns encrypt with different encryption scheme for example the column “rand” become Col1OnionEq, Col1OnionOrder and Col1OnionSearch.

Example:

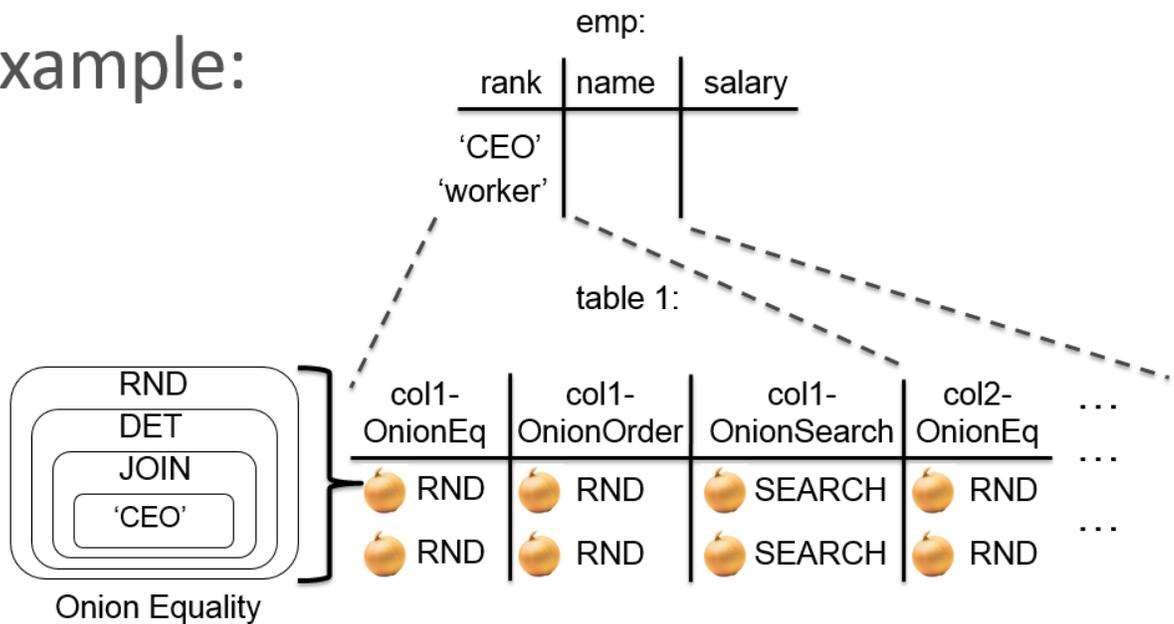


Figure 4 three columns example in CryptDB Popa, Redfield, Zeldovich, & Balakrishnan (2011).

3.3 Layered architecture

A layered architecture defines interfaces between codes that implements the various concerns, allowing a change to the way one concern is implemented without significant disruption to code in the other layers. The rules are as follows:

- Layers communicate top to bottom. A layer is dependent only on the layer directly below it.
- Each layer is unaware of any other layers except for the layer just below it.

Typically, application architecture uses three layers Bauer & King(2004):

- **Presentation layer:**

The user interface logic is topmost. Code responsible for the presentation and control of page and screen navigation forms the presentation

- **Business logic layer:**

The exact form of the next layer varies widely between applications. It is generally agreed, however, that this business layer is responsible for implementing any business rules or system requirements that would be understood by users as part of the problem domain.

- **Persistence layer:**

The persistence layer is a group of classes and components responsible for data storage to, and retrieval from, one or more data stores.

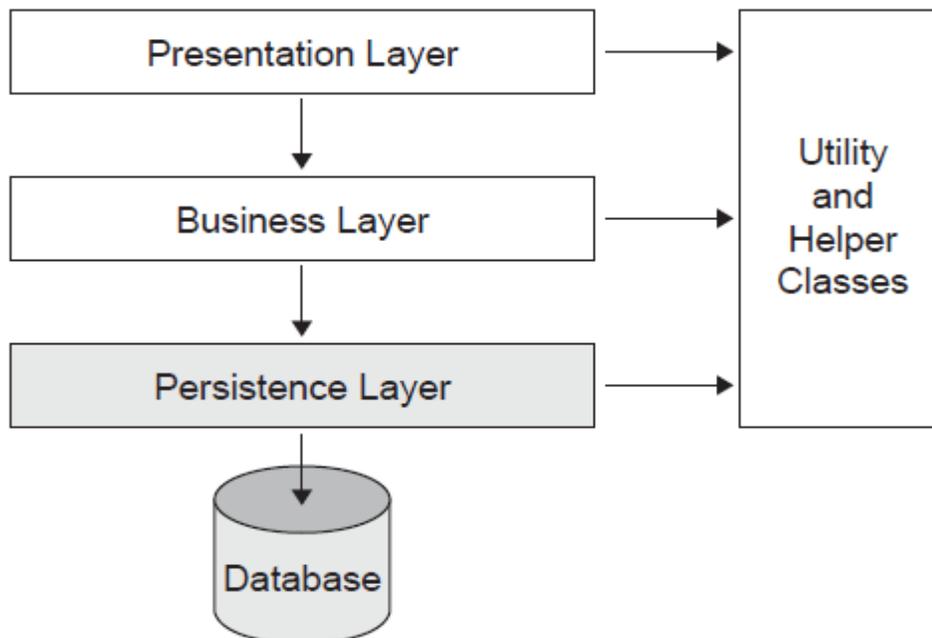


Figure 5 application architecture layers Bauer & King(2004).

3.4 Object Relational Mapping (ORM) Overview

Concisely, object-relational mapping is the automated and transparent persistence of the objects in the application to the tables in a relational database. In another word, it is the middleware between the objects and the tables that manages persistence Bauer & King (2004)

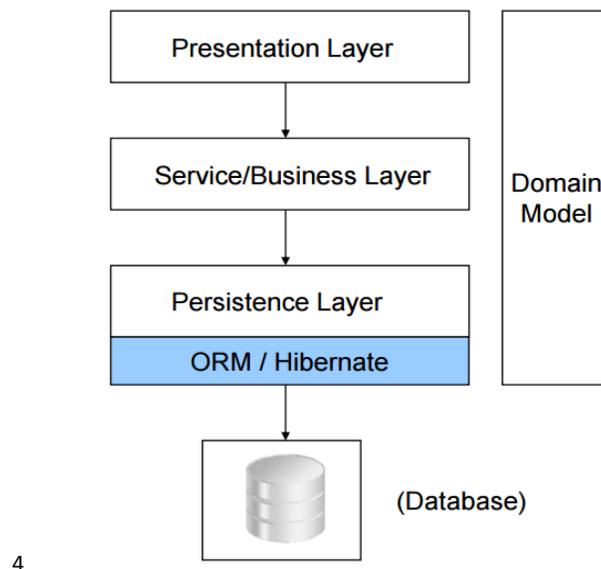


Figure 6 application architecture layers with ORM Bauer & King(2004).

3.4.1 What do we mean by persistence?

Persistence is one of the fundamental concepts in application development. By persistence, we mean preserving the data that entered from the users; otherwise, we are going to lose all the data when the host machine will power off Bauer & King(2004)

Advantages of ORM Bauer & King (2004):

- **Productivity**

- Eliminates lots of repetitive code
- focus on business logic.
- Database schema is generated.

- **Maintainability**

- Fewer lines of code.
- Easier to understand.
- Easier to the manage change in the object model.

- **Performance**

- Lazy loading.
- Associations are fetched when needed.
- Caching.

- **Database vendor independence**

- The underlying database is abstracted away.
- Can be configured outside the application

3.4.2 ORM Mapping example:

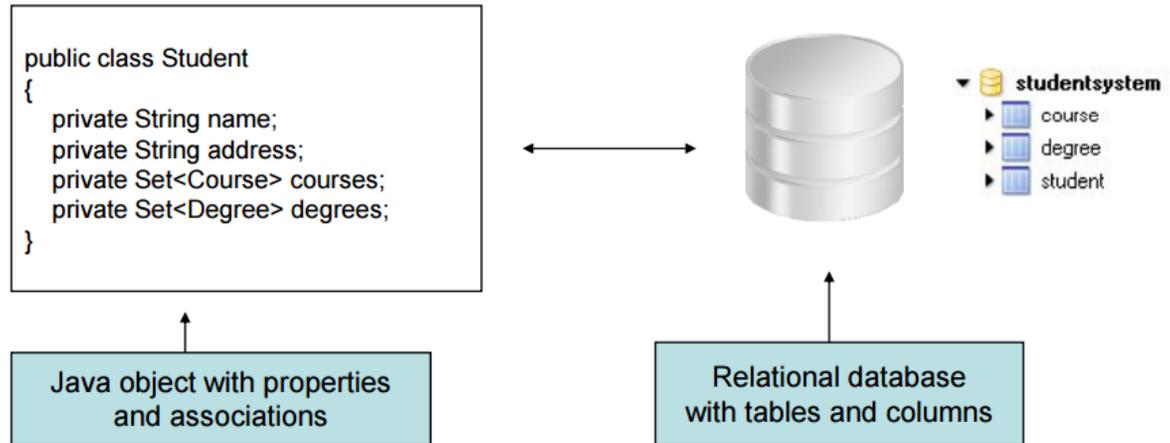


Figure 7 ORM Mapping example Bauer & King(2004).

3.4.3 How to solve the mismatching between the tables in the relational database and the classes in the object oriented.

'Object-Relational Impedance Mismatch' (sometimes called the 'paradigm mismatch') is just a fancy way of saying that object models and relational models do not work very well together. RDBMSs represent data in a tabular format (a spreadsheet is a good visualization for those not familiar with RDBMSs), whereas object-oriented languages, such as Java, represent it as an interconnected graph of objects. Loading and storing graphs of objects using a tabular relational database exposes us to five mismatch problems...

a. Subtypes (inheritance):

ORM enables you to develop persistent classes following natural Object-oriented idioms including inheritance, polymorphism, association, composition, for example:

Suppose that you have Employee & Customer Classes, which has derived from User Class, now the problem how to map this classes structure with the relational database, because in the relational database there is no concept of inheritance, also the same apply for the polymorphism, in the object orientate we have base class & sub class which we can override it, but in the relational database there is no such notion that exist in the relational database.

b. Granularity:

Sometimes you will have an object model, which has more classes than the number of corresponding tables in the database (we says the object model is more granular than the relational model). Take for example the notion of an Address where there is only one table called “User” but in the object oriented design there are two classes first one called “User” and the second called “Address”, so the object relational mapper has to handle this mismatch where the number of tables mapped to different number of classes.

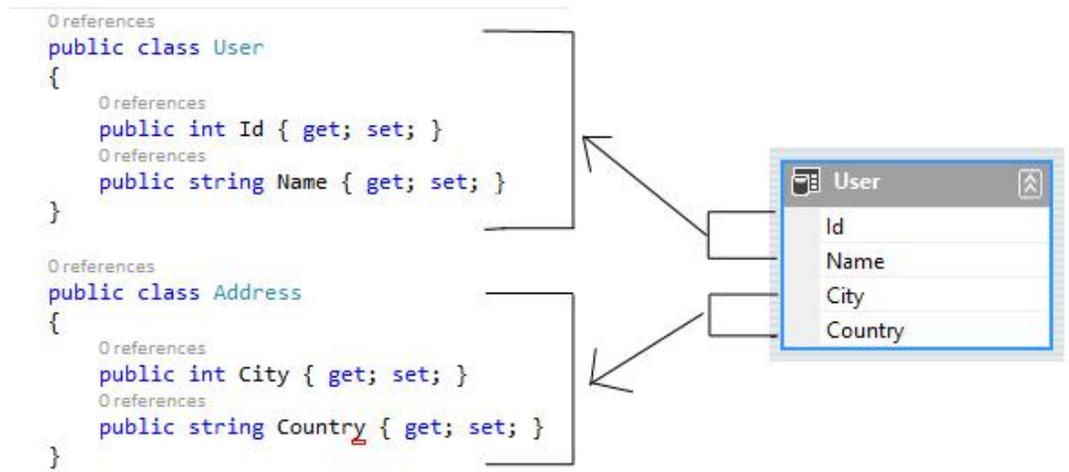


Figure 8 Granularity example.

c. Identity:

A RDBMS defines exactly one notion of 'sameness': the primary key. Some OO languages like Java, defines both object identity $a==b$ and object equality $a.equals(b)$.

d. Associations:

Associations are represented as unidirectional references in Object Oriented languages whereas RDBMSs use the notion of foreign keys. If you need bidirectional relationships in some OO languages like Java, you must define the association twice.

Likewise, you cannot determine the multiplicity of a relationship by looking at the object domain model.

e.Data navigation:

The way you access data in Java is fundamentally different than the way you do it in a relational database. In Java, you navigate from one association to another walking the object network.

This is not an efficient way of retrieving data from a relational database. You typically want to minimize the number of SQL queries and thus load several entities via JOINS and select the targeted entities before you start walking the object network.

3.5 Choosing our ORM Library

To prove this idea, we have implemented our POC (proof of concept) framework, This framework build on previous framework which already developed to cover all ORM concepts, simultaneously it should be open source library to allow us make our changes.

There are many of ORM open source frameworks, in this thesis we are not going to discuss which perform better, just to prove this idea we have chosen NHibernate as our experimental ORM. Note that this idea can be implemented in any open source ORM framework.

3.6 Implementation

We are going to use the following simple example to explain our implementation.

In C# code deliver a query written in SQL, to a database engine.

```
string sql = "SELECT ... FROM employee WHERE id = 10";
SqlCommand cmd = new SqlCommand(connection, sql);
Result res = cmd.Execute();
string email = res[0]["email"];
```

In contrast, the following makes do the same of the previous C# code, but using an ORM API.

```
Employee emp = repository.GetEmployee(10);
string email = emp.getEmail ();
```

There are many issues must be considered while we are implementing our new framework, especially on merging ORM framework With CryptDB project:

- How to interrupt real time SQL queries that generated from object relational mapper.

As we sow in the previous example:

```
Employee emp = repository.GetEmployee(10);
```

Object relational mapper will automatically take care of generating the proper SQL query to get the employee who has id 10, the programmer will not write that SQL by himself.

The new framework solved that issue, by intercept the auto generated queries then applying CryptDB security concepts.

- How to rewrite (encrypt) database schema, to incomprehensible schema.

After interrupting the auto generated SQL, especially in Data Definition Language, we are going to replace each column with three incomprehensible columns, those columns representing the same onions in the CryptDB project.

- Column Onion Equality.
- Column Onion Order.
- Column Onion Search if the value is text OR Homomorphic if the value is number.

Example:

Auto generated table like

```
Create table Employee (ID integer, Name text)
```

Output after applying CryptDB concepts

```
Create table Tb11 (Col1_Eq text, Col1_Order text , Col1_Hom text, Col2_Eq text , Col2_Order, text Col2_Search)
```

Real example for creating table In CryptDB project:

```

azureuser@MYMEUubuntu1204: ~
ENCRYPTE RESULTS:
+
|
+
+
=====
QUERY: create table EmployeeTbl (ID text , Email text , Mobile text)
NEW QUERY: create table table_LWMVPLOMFH (DSBOJCJVCQoEq BLOB, HNNVJEHZSooOrder B
IGINT unsigned, cdb_saltZPPFEISUHS BIGINT(8) unsigned, MHINVUQGRUoEq BLOB, WBILW
TQDSIoOrder BIGINT unsigned, cdb_saltISDXZIHREQ BIGINT(8) unsigned, YFSIVPCNVWoE
q BLOB, HUXPFTIVZCoOrder BIGINT unsigned, cdb_saltSAAXAAHHTL BIGINT(8) unsigned)
  AUTO_INCREMENT=0 ENGINE=InnoDB
ENCRYPTE RESULTS:
+
|
+
+
ENCRYPTE RESULTS:
+
|
+
+
t

```

Figure 9 Creating table in CryptDB project

From the previous figure, we can note how CryptDB project encrypt table name from “EmployeeTbl” to a new name which not understandable, also note that each column become three columns those three columns represent the three onions (Col1OnionEq, Col1OnionOrder and Col1OnionSearch).

- How to rewrite (encrypt) the plain SQL that generated from the ORM, to incomprehensible SQL.

After we encrypted the database schema, including the column name in the previous step, our new framework will deal with the new column name.

Example:

```
Select ID, Name From Employee
```

Output after applying CryptDB concepts

```
Select Col1_Hom, Col2_Search From Tb11
```

- How to rewrite (encrypt) the values inside the SQL queries to incomprehensible values.

Example:

```
where ID = 3
```

Output after applying CryptDB concepts

```
where Col1 = xyz. --where the xyz is the encryption stake.
```

- How to choose and process (decrypt if needed) the correct column based on the coming query.

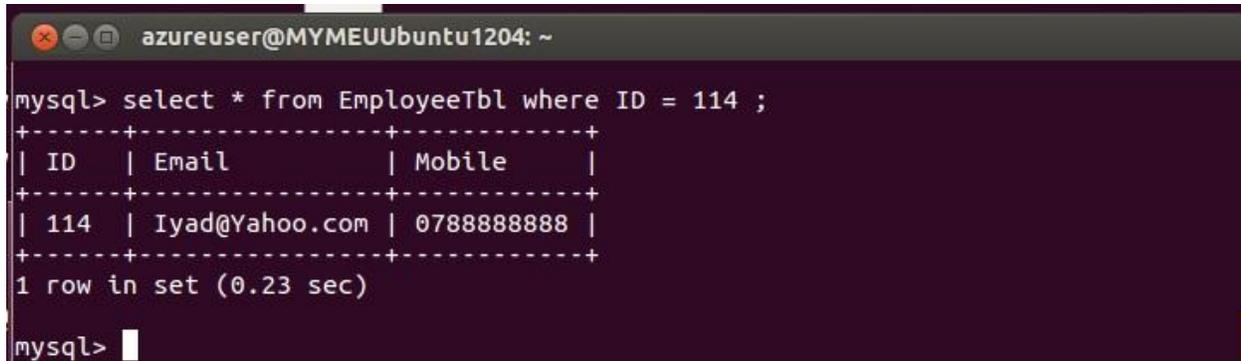
Example1:

```
Select * From Employee where ID = 3
```

Output after applying CryptDB concepts

```
Select * From Tbl1 where Col1_Eq = xyz
```

Real example for select data from table with equality parameter:



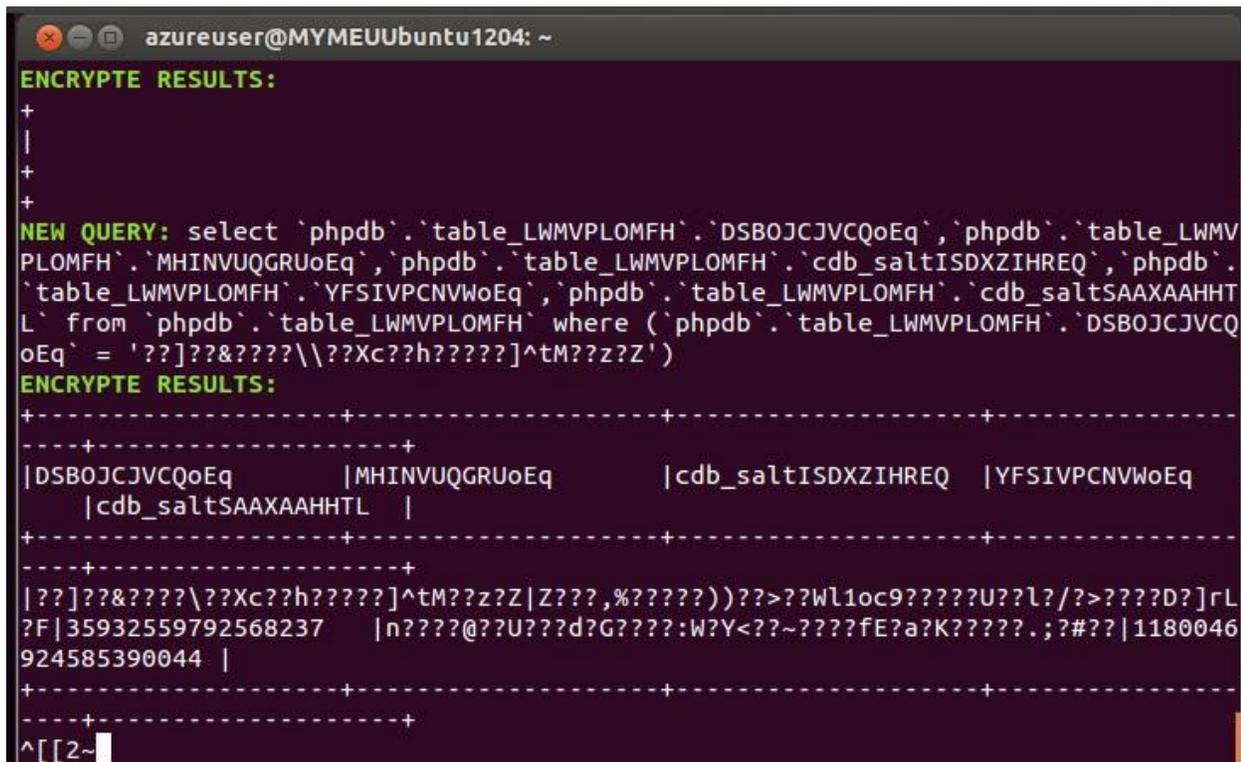
```

azureuser@MYMEUUbuntu1204: ~
mysql> select * from EmployeeTbl where ID = 114 ;
+-----+-----+-----+
| ID   | Email           | Mobile       |
+-----+-----+-----+
| 114  | Iyad@Yahoo.com  | 0788888888  |
+-----+-----+-----+
1 row in set (0.23 sec)

mysql>

```

Figure 10 Trusted server in CryptDB project 1



```

azureuser@MYMEUUbuntu1204: ~
ENCRYPTE RESULTS:
+-----+-----+-----+
|
|
+-----+-----+-----+
NEW QUERY: select `phpdb`.`table_LWMVPLOMFH`.`DSB0JCJVCQoEq`, `phpdb`.`table_LWMV
PLOMFH`.`MHINVUQGRUoEq`, `phpdb`.`table_LWMVPLOMFH`.`cdb_saltISDXZIHREQ`, `phpdb`.`
table_LWMVPLOMFH`.`YFSIVPCNVwoEq`, `phpdb`.`table_LWMVPLOMFH`.`cdb_saltSAAXAAHHT
L` from `phpdb`.`table_LWMVPLOMFH` where (`phpdb`.`table_LWMVPLOMFH`.`DSB0JCJVCQ
oEq` = '??]??&????\??Xc??h????]^tM??zZ')
ENCRYPTE RESULTS:
+-----+-----+-----+
|DSB0JCJVCQoEq      |MHINVUQGRUoEq      |cdb_saltISDXZIHREQ  |YFSIVPCNVwoEq
|cdb_saltSAAXAAHHTL |
+-----+-----+-----+
|??]??&????\??Xc??h????]^tM??zZ|Z???,%?????)??>??Wl1oc9?????U??l/?>????D?]rL
?F|35932559792568237 |n????@??U??d?G????:W?Y<??~????fE?a?K?????.;#??|1180046
924585390044 |
+-----+-----+-----+
^[[2~

```

Figure 11 Untrusted server in CryptDB project 1

From the previous figure, we can note how CryptDB project rewrite the query to be compatible with the encrypted schema, first encrypt table name from “EmployeeTbl” to the same name which encrypted when database created, also note that each column have been encrypted to be compatible with the columns in the encrypted database, and because they're “where id = 114” the system choose the Col1OnionEq to do that matching, not and Col1OnionOrder or Col1OnionSearch cause it's equality operation.

Example2:

```
Select * From Employee order by ID
```

Output after applying CryptDB concepts

```
Select * From Tb11 order by Col1_Order
```

Real example for select data from table with Order By parameter:



```
azureuser@MYMEUubuntu1204: ~
mysql> Select * from EmployeeTbl order by ID ;
+-----+-----+-----+
| ID   | Email           | Mobile   |
+-----+-----+-----+
| 114  | Iyad@Yahoo.com | 0788888888 |
+-----+-----+-----+
1 row in set (1.17 sec)

mysql>
```

Figure 12 Trusted server in CryptDB project 2

```

azureuser@MYMEUubuntu1204: ~
ENCRYPTE RESULTS:
+
|
+
+
NEW QUERY: select `phpdb`.`table_LWMVPLOMFH`.`DSBOJJCVCQoEq`,`phpdb`.`table_LWMV
PLOMFH`.`cdb_saltZPPFEISUHS`,`phpdb`.`table_LWMVPLOMFH`.`MHINVUQGRUoEq`,`phpdb`.`
`table_LWMVPLOMFH`.`cdb_saltISDXZIHREQ`,`phpdb`.`table_LWMVPLOMFH`.`YFSIVPCNVWoE
q`,`phpdb`.`table_LWMVPLOMFH`.`cdb_saltSAAXAAHHTL` from `phpdb`.`table_LWMVPLOMF
H` order by `phpdb`.`table_LWMVPLOMFH`.`HNNVJEHZS0oOrder`
ENCRYPTE RESULTS:
+-----+-----+-----+-----+
|DSBOJJCVCQoEq      |cdb_saltZPPFEISUHS |MHINVUQGRUoEq      |cdb_saltISDXZIHREQ
EQ |YFSIVPCNVWoEq      |cdb_saltSAAXAAHHTL |
+-----+-----+-----+-----+
|>???T????????56?R?Gm?AT?!?YF?????????7??????????|5098228596690293409 |Z???,%???(
??)??>??Wl1oc9?????U??l?/?>????D?]rL?F|35932559792568237 |n????@??U??d?G????
:W?Y<??~????fE?a?K?????.;?#??|1180046924585390044 |
+-----+-----+-----+-----+
^[[2~^[[2~^[[2~

```

Figure 13 Untrusted server in CryptDB project 2

Real example for inserting data in CryptDB project:

```

azureuser@MYMEUubuntu1204: ~
+
|
+
+
t^[[2~=====
QUERY: insert into EmployeeTbl (ID , Email , Mobile) values ("114" , "Iyad@Yahoo
o.com" , "0788888888" )
NEW QUERY: insert into `phpdb`.`table_LWMVPLOMFH` (`phpdb`.`table_LWMVPLOMFH`.`D
SBOJJCVCQoEq`,`phpdb`.`table_LWMVPLOMFH`.`HNNVJEHZS0oOrder`,`phpdb`.`table_LWM
VPLOMFH`.`cdb_saltZPPFEISUHS`,`phpdb`.`table_LWMVPLOMFH`.`MHINVUQGRUoEq`,`phpd
b`.`table_LWMVPLOMFH`.`WBILWTQDSIoOrder`,`phpdb`.`table_LWMVPLOMFH`.`cdb_saltIS
DXZIHREQ`,`phpdb`.`table_LWMVPLOMFH`.`YFSIVPCNVWoEq`,`phpdb`.`table_LWMVPLOMFH
`.`HUXPFTIVZCoOrder`,`phpdb`.`table_LWMVPLOMFH`.`cdb_saltSAAXAAHHTL`) values ('
>???T????????56?R?Gm?AT?!?YF?????????7??????????\Z?', 5490451521927050552, 5098228
596690293409, 'Z???,%?????)??>??Wl1oc9?????U??l?/?>????D?]rL?F', 88021721792462
37149, 35932559792568237, 'n????@??U??d?G????:W?Y<??~????fE?a?K?????.;?#??', 83
51654101297750855, 1180046924585390044)
ENCRYPTE RESULTS:
+
|
+
+

```

Figure 14 Inserting data in CryptDB project

Real example for updating row in CryptDB project:

```

azureuser@MYMEUubuntu1204: ~
QUERY: phpdb
NEW QUERY: USE `phpdb`
=====
QUERY: [00]
unexpected packet type 27
=====
QUERY: UPDATE EmployeeTbl SET Email = 'Nain@gmail.com' , Mobile = 'MyP@ss' WHERE
ID = '210'
NEW QUERY: update (`phpdb`.`table_LWMVPLOMFH`) set `phpdb`.`table_LWMVPLOMFH`.`M
HINVUQGRUoEq`='\|=????????\T?\r???h)?i???Q?H????-c?h??\Zbc8??T?9-?', `phpdb`.`t
able_LWMVPLOMFH`.`WBILWTQDSIoOrder`=3110651451326483373, `phpdb`.`table_LWMVPLOM
FH`.`cdb_saltISDXZIHREQ`=8915532859755687807, `phpdb`.`table_LWMVPLOMFH`.`YFSIVP
CNVWoEq`='&??]?@?????K?9?e?~????}??C??_??+?9????????=o????m', `phpdb`.`table_LWMV
PLOMFH`.`HUXPFTIVZCoOrder`=1412206671873991439, `phpdb`.`table_LWMVPLOMFH`.`cdb_
saltSAAXAAHHTL`=10705505351491193857 where (`phpdb`.`table_LWMVPLOMFH`.`DSBOJCJV
CQoEq` = '??_[att????jo?5-???[kX???QO??{??p')
ENCRYPTE RESULTS:
+
|
+
+
=====
QUERY:
^[[2~

```

Figure 15 Creating table in CryptDB project

Real example for deleting table in CryptDB project:

```

azureuser@MYMEUubuntu1204: ~
ENCRYPTE RESULTS:
+
|
+
+
=====
QUERY: create table EmployeeTbl (ID text , Email text , Mobile text)
NEW QUERY: create table table_LWMVPLOMFH (DSBOJCJVCQoEq BLOB, HNNVJEHZSooOrder B
IGINT unsigned, cdb_saltZPPFEISUHS BIGINT(8) unsigned, MHINVUQGRUoEq BLOB, WBILW
TQDSIoOrder BIGINT unsigned, cdb_saltISDXZIHREQ BIGINT(8) unsigned, YFSIVPCNVWoE
q BLOB, HUXPFTIVZCoOrder BIGINT unsigned, cdb_saltSAAXAAHHTL BIGINT(8) unsigned)
AUTO_INCREMENT=0 ENGINE=InnoDB
ENCRYPTE RESULTS:
+
|
+
+
ENCRYPTE RESULTS:
+
|
+
+
t

```

Figure 16 Deleting table in CryptDB project

CHAPTER FOUR

The Discussion of Results

4.1 Introduction

In this chapter, we are going to analysis and compare the result, between the original CryptDB project and the proposed development framework using NHibernate, Comparing done using two main development languages C# .NET and PHP, furthermore we have added some simple code for the CRUD operations to make the comparison clearer.

4.2 Comparing the result with the previous work (CryptDB)

To prove our thesis idea, we have created a simple database, consist from Employee table which has 3 columns: ID, Email and Mobile. Then we have developed three applications doing same CRUD operations in three ways:

- Original CryptDB project using PHP language on Ubuntu.
- Original CryptDB project using ADO.NET with C# language on Windows.
- Proposed development framework Using NHibernate on Windows. .

Usually the Application consists of several layers like presentation layer, Business Logic Layer and Data Access Layer, this code sample will focus on the last layer (DAL) because it has the code to retrieve and modify data from the database.

We are going to discuss the result after implementing the following scenarios:

Add New Row

Email :

Mobile :

```
// Add new Employee with the original CryptDB Project using PHP language.
function AddEmployee($pEmployee)
{
    $sql = "INSERT INTO EmployeeTbl "(ID , Email , Mobile) ".
"VALUES('$pEmployee->ID','$pEmployee->Email','$pEmployee->Mobile')";
    $result = mysql_query( $sql, $this->conn);
}
```

```
// Add new Employee with the original CryptDB Project using C# language.
public void AddEmployee(Employee emp)
{
    MySqlCommand cmd = new MySqlCommand("INSERT INTO
Employee(Email,Mobile) values( " + emp.Email + " , " + emp.Mobile + " )",
cn);
    cmd.ExecuteNonQuery();
}
```

```
// Add new Employee with The proposed development framework Using
NHibernate.
public void AddEmployee(Employee pEmployee)
{
    session.Save(pEmployee);
    transaction.Commit();
}
```

In the previous scenario we have three different functions:

- First one written using PHP language, this function works with the original CryptDB project, the main idea was to insert a new row, and because we are using the original CryptDB we have to write the insert SQL statement explicitly.
- Second one written using ADO C# .NET, this function works also with the original CryptDB project, the main idea was to insert a new row using new language to show that the original CryptDB can work with different development languages, but still we have to run CryptDB proxy in Ubuntu, and also because we are using the original CryptDB we have to write the insert SQL statement explicitly.
- Last one show our proposed development framework, our “AddEmployee” function was written using the same language which used in the last function (C# .NET), but here because we are using an object relational mapping technique (Nhibernate). There is no need to write the insert SQL statement explicitly, our save function will create the proper SQL which will be like

```
INSERT INTO XYZ_Table ( Column1_Eq , Column1_Ord , Column1_Search ,  
Column2_Eq , Column2_Ord , Column2_Search) values(  
Encrypted_Val_Column1_Eq , Encrypted_Val_Column1_Ord ,  
Encrypted_Val_Column1_Search , Encrypted_Val_Column2_Eq ,  
Encrypted_Val_Column2_Ord , Encrypted_Val_Column2_Search)
```

ID :

Update Row

Email : Mobile:

```
//update existing Employee with the original CryptDB Project using PHP
language.
function UpdateEmployee ($pEmployee)
{
    $sql = "UPDATE EmployeeTbl SET ". "Email = '$pEmployee->Email' ,
Mobile = '$pEmployee->Mobile' ". "WHERE ID = '$pEmployee->ID'";
    $result = mysql_query( $sql, $this->conn);
}
```

```
//update existing Employee with the original CryptDB Project using C#
language.
public void UpdateEmployee(Employee emp)
{
    MySqlCommand cmd = new MySqlCommand("UPDATE Employee SET Email = "
+ emp.Email + " , Mobile = " + emp.Mobile + " where ID = " + emp.ID + " ",
cn);
    cmd.ExecuteNonQuery();
}
```

```
//update existing Employee with The proposed development framework Using
NHibernate.
public void UpdateEmployee (Employee pEmployee)
{
    session. Update(pEmployee);

    transaction.Commit();
}
```

Delete Row

ID :

Delete

```
//Delete existing Employee with the original CryptDB Project using PHP
language.
function DeleteEmployee ($pEmployee)
{
    $sql = "Delete FROM EmployeeTbl WHERE ID = '$pEmployee->ID'";
    $result = mysql_query( $sql, $this->conn);
}
```

```
//Delete existing Employee with the original CryptDB Project using C#
language.
public void DeleteEmployee(int p_Emp_ID)
{
    MySqlCommand cmd = new MySqlCommand("Delete Employee where ID = " +
p_Emp_ID + " ", cn);
    cmd.ExecuteNonQuery();
}
//Delete existing Employee with The proposed development framework Using
NHibernate.
public void DeleteEmployee(Employee pEmployee)
{
    session.Delete(pEmployee);
    transaction.Commit();
}
```

Select All Rows in the table

```
//Select All Employees with the original CryptDB Project using C# language.
public IList<Employee> Get_All_Employees()
{
    DataTable dt_Result = new DataTable();
    IList<Employee> result = new List<Employee>();
    MySqlCommand cmd = new MySqlCommand("Select * from Employee", cn);
    cmd.ExecuteNonQuery();
    MySqlDataAdapter adp = new MySqlDataAdapter(cmd);
    adp.Fill(dt_Result);
    // convert SQL result to object, this is the manual way, we can use
    SQL mapping library like (Linq or iBatis)
    if (dt_Result != null && dt_Result.Rows.Count > 0)
    {
        for (int i = 0; i < dt_Result.Rows.Count; i++)
        {
            result.Add(new Employee { Email =
dt_Result.Rows[i]["Email"].ToString(), Mobile =
dt_Result.Rows[i]["Mobile"].ToString() });
        }
    }
    return result;
}
```

```
//Get All Employees with The proposed development framework Using NHibernate.
public IList<Employee> Gell_All_Employee()
{
    return session.CreateCriteria<Employee>().List<Employee>();
}
```

Comparing the result, between the original CryptDB project and the proposed development framework:

- The original CryptDB project Works only with MySQL Database, while the proposed development framework Works with all relational databases like MySQL, Oracle and SQL server.
- The original CryptDB project Work only with Ubuntu, because it has proxy which already developed on Ubuntu, while the proposed development framework Work with any operating system, because it's a portable DLL file.
- In the original CryptDB project the developer must create database by himself, while the database will be created automatically, and the developer will not create the database anymore. In the proposed development framework.
- In the original CryptDB project the developer must write SQL by himself, while in the proposed development framework the SQL will be written automatically, and the developer will not write SQL anymore.
- The proposed development framework will enhance of the developer productivity since it will take less development time (no need SQL).

4.3 Contribution:

Propose first Object Relational Mapper works with encrypted database, usually Object Relational Mapper help the developers during development stage, by eliminating a lot of repetitive logic, ORM creates that repetitive logic (SQL queries) especially in Create, Read, Update, and Delete operations, the proposed development framework will encourage the developers to develop secure applications instead of developing unsecured applications.

4.4 Work Limitations:

The current proposed development framework based on the CryptDB work, and we have implemented only the basic operations (create, read, update & delete), here we are going to talk about the major limitations such as the original ORM can handle both user defined function and stored procedure while the proposed development framework will not handle them. Another limit that the encryption stack for the data become weaker over time since we are going to decrypt data layer based on the coming query, and this may help the hacker ho hack the weaker column between the three columns. Another limit the joins between tables which used to combine rows from two or more tables, this is not handled in the work.

CHAPTER FIVE

Conclusions and Future Work

5.1 Introduction

Nowadays, there are many applications store their data on servers, and a lot of them used the cloud to host those servers, this led to the widespread confidential data theft. One of the most popular and practical projects that already protected data confidentiality even when attackers get access to all server data called “CryptDB”, a lot of companies or organizations used or adopted CryptDB, for example Google has developed an experimental extension of the BigQuery client, known as Encrypted BigQuery, which was informed and motivated by the CryptDB paper. Another example, SAP AG developed a system called SEED, which implements CryptDB's design on top of their HANA database system Popa Raluca Ada (2014).

Another popular programming technique in the software engineering called “Object-relational mapping”, ORM is a technique that lets programmer query and manipulate data from a database using an object-oriented paradigm without writing SQL manually.

5.2 Conclusions

In this thesis, we have discussed how to merge and restructure CryptDB project and ORM framework for the basic operations insert, update, delete and read. The resulted development framework works with all relational databases, in contrast CryptDB work only with MySQL database, reducing the required time in the development stage cause the developer will not wright SQL manually.

5.3 Recommendations for Future Work

Based on the reported work, the following suggestions can be put forward for future research work to get a more practical development framework:

- Joins between tables which used to combine rows from two or more tables.
- Projection (select columns with alias from different tables).
- Subquery (Inner query or Nested query) which is query within another SQL query and embedded within the WHERE clause.
- Restrictions (Disjunction & Conjunction), used in the ORM like AND & OR in the SQL.

References

- Amanatidis, G., Boldyreva, A., & O'Neill, A. (2007). New security models and provably-secure schemes.
- Bauer, C., & King, G. (2004). Hibernate in action. Manning publications co.
- Evdokimov, S., & Günther, O. (2007). Encryption techniques for secure database outsourcing.
- Fast, secure encryption for indexing in a column-oriented. (2007).
- Gentry, C. (2009). A fully homomorphic encryption scheme.
- Popa, R. A., H. Li, F., & Zeldovich, N. (2013). An ideal-security protocol for order-preserving encoding. the 34th IEEE Symposium on Security and Privacy (IEEE S&P/Oakland).
- Popa, R., Redfield, C., Zeldovich, N., & Balakrishnan, H. (2011). CryptDB: protecting confidentiality Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP).
- Rani, D., & Ranjan, R. K. (2014). A Comparative study of saaS, paaS and iaaS in cloud computing. International Journal of Advanced Research in Computer Science and Software Engineering.
- Sathyavani, M. (2013). Survey on Cloud Computing. International Journal of Computer Trends and Technology (IJCTT).

- Tu, S., Kaashoek, M., Madden, S., & Zeldovich, N. (2013). Processing Analytical Queries over Encrypted Data. the 39th International Conference on Very Large Data Bases (VLDB).
- What Is an ORM. (n.d.). Retrieved from hibernate (On-Line), available: <http://hibernate.org/orm/what-is-an-orm/>.
- Jaroslav, O. (2006). Object-relational mapping (Unpublished diploma thesis), Comenius University, Bratislava, Slovakia
- David, T. B. (2014). Information Systems for Business and Beyond.
- Velumadhava, R. & Selvamanib, K. (2015) Data security challenges and its solutions in cloud computing International Conference on Computer, Communication and Convergence, 204–209
- Sheriff, R. (Ed.). (2014). Electronics and Telecommunications Research Seminar Series. School of Electrical Engineering & Computer Science University of Bradford. Bradford. UK
- Sabahi, F. (2011) Cloud computing security threats and responses. In Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on, 2011, 245-249.
- Jeffrey, U. & Alfred, A., (1992) Foundations of computer science (C Edition)
- Michael, J. H. (2013) Database Design for Mere Mortals (3rd Edition)

- Microsoft, Corp. Types of table relationships (Visual database tools) , (On-Line), available: [https://technet.microsoft.com/en-us/library/ms190651\(v=sql.105\).aspx](https://technet.microsoft.com/en-us/library/ms190651(v=sql.105).aspx)
- Manoj, K., Ravinder, T. & Manish, M. (2014) A Tool for quality measurement of software based on object oriented design. International Journal of Advance Research in Computer Science and Management Studies Volume 2, Issue 5, May 2014 pg. 140-144
- William S. D. & David C. Y. (1998). The Information System Consultant's Handbook. Boca Raton, London, New York and Washington.
- Lopes, C. C., Times, V. C., Matwin, S., Ciferri, R. R. & Ciferri, C. D. (2014) *Processing OLAP queries over an encrypted data warehouse stored in the cloud* In Data Warehousing and Knowledge Discovery.
- Cao, N., Labs, W., View, M., Wang, C., Li, M. & Ren, K. (2014) Privacy-preserving multi-keyword ranked search over encrypted cloud data, *Parallel and Distributed Systems, IEEE Transactions* P 222 – 233
- Wang, B., Virginia Polytech. Inst. & State Univ., Yu, S., Lou, W. & Hou, Y.T. (2014) Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, *INFOCOM, 2014 Proceedings IEEE* P 2112 – 2120.
- Popa Raluca Ada (2014). **Building practical systems that compute on encrypted data.** (Unpublished doctoral dissertation) Massachusetts Institute of Technology, United States.

- Popa, R. A., Stark. E., Helfer, J., Valdez, .S, Zeldovich, N., Kaashoek, M. F., & Balakrishnan, H. (2014). Building web applications on top of encrypted data using Mylar. **MIT CSAIL and &Meteor Development Group.**
- Shen, Z., Dept. of Computer. Sci. & Technol., Tsinghua Univ., Beijing, China, Shu, J. & Xue, W. (2013) Preferred keyword search over encrypted data in cloud computing. *Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium.*
- Veitch, A. (2002) *Relational database connectivity* In *Zope2 book.*