Master of Information Technology (Research) Thesis
IMPROVING TRUST AND SECURING DATA ACCESSIBILITY FOR E-HEALTH DECISION MAKING BY USING DATA ENCRYPTION TECHNIQUES

Dong Il Shin
Master of Information Technology (Research)

Submitted in fulfilment of the requirements for the degree of
Master of Information Technology (Research)

Computer Science Discipline
Faculty of Science and Technology
Queensland University of Technology

January 2012
Keywords

AES (Advanced Encryption Standard), Bloom filter, Bucket index, Collection classes, DAS (Data as Service), Encryption, LINQ (Language Integrated Query), Microsoft Built-in DBMS (Database Management System), Outsourcing, Partitioning, Singleton design pattern, Symmetric key Encryption
Abstract

In the medical and healthcare arena, patients’ data is not just their own personal history but also a valuable large dataset for finding solutions for diseases. While electronic medical records are becoming popular and are used in healthcare work places like hospitals, as well as insurance companies, and by major stakeholders such as physicians and their patients, the accessibility of such information should be dealt with in a way that preserves privacy and security. Thus, finding the best way to keep the data secure has become an important issue in the area of database security. Sensitive medical data should be encrypted in databases. There are many encryption/ decryption techniques and algorithms with regard to preserving privacy and security. Currently their performance is an important factor while the medical data is being managed in databases. Another important factor is that the stakeholders should decide more cost-effective ways to reduce the total cost of ownership. As an alternative, DAS (Data as Service) is a popular outsourcing model to satisfy the cost-effectiveness but it takes a consideration that the encryption/ decryption modules needs to be handled by trustworthy stakeholders.

This research project is focusing on the query response times in a DAS model (AES-DAS) and analyses the comparison between the outsourcing model and the in-house model which incorporates Microsoft built-in encryption scheme in a SQL Server. This research project includes building a prototype of medical database schemas. There are 2 types of simulations to carry out the project. The first stage includes 6 databases in order to carry out simulations to measure the performance between plain-text, Microsoft built-in encryption and AES-DAS (Data as Service). Particularly, the AES-DAS incorporates implementations of symmetric key encryption such as AES (Advanced Encryption Standard) and a Bucket indexing processor using Bloom filter. The results are categorised such as character type, numeric type, range queries, range queries using Bucket Index and aggregate queries. The second stage takes the scalability test from 5K to 2560K records.

The main result of these simulations is that particularly as an outsourcing model, AES-DAS using the Bucket index shows around 3.32 times faster than a normal AES-DAS under the 70 partitions and 10K record-sized databases. Retrieving Numeric typed data takes shorter time than Character typed data in AES-DAS. The aggregation query response time in AES-DAS is not as consistent as that in MS built-in encryption scheme. The scalability test shows that the DBMS reaches in a certain threshold; the query response time becomes rapidly slower. However, there is more to investigate in order to bring about other outcomes and to construct a secured EMR (Electronic Medical Record) more efficiently from these simulations.
Table of Contents

Keywords .......................................................................................................................... i
Abstract ........................................................................................................................... ii
Table of Contents ........................................................................................................... iii
List of Figures .................................................................................................................... v
List of Tables .................................................................................................................... vii
List of Publications ........................................................................................................ viii
List of Abbreviations ...................................................................................................... ix
Statement of Original Authorship .................................................................................. x
Acknowledgments ........................................................................................................... xi

1 CHAPTER 1: INTRODUCTION ...................................................................................... 13
1.1 Background ............................................................................................................... 13
1.2 Context ..................................................................................................................... 14
1.3 Purposes ................................................................................................................... 14
1.4 Significance .............................................................................................................. 15
1.5 Thesis Outline ......................................................................................................... 16
1.6 Summary .................................................................................................................. 17

2 CHAPTER 2: LITERATURE REVIEW ......................................................................... 18
2.1 Introduction to medical IT solutions including EMR and EHR ............................. 18
2.2 Introduction to Current Cryptography Technologies ......................................... 27
2.3 Bucket Index, Bloom Filter and Related issues .................................................. 30
2.4 DAS model and Query Processor ......................................................................... 34
2.5 Number theory ....................................................................................................... 39
2.6 Extensible Software Frameworks for EMRs ......................................................... 40
2.7 Introduction to Built-in DBMS Security ............................................................... 44
2.8 Review of Aggregate-based Query ....................................................................... 46
2.9 Implications ........................................................................................................... 48
2.10 Summary ............................................................................................................... 48

3 CHAPTER 3: RESEARCH DESIGN ........................................................................... 49
3.1 Introduction .............................................................................................................. 49
3.2 Methodology .......................................................................................................... 50
3.2.3 Stage 2: Simulation Procedure ....................................................................... 55
3.3 Instruments ............................................................................................................. 58
3.4 Analysis ................................................................................................................... 58
3.5 Applications for the Simulations .......................................................................... 59
3.6 Summary ............................................................................................................... 65
## 4 CHAPTER 4: SIMULATIONS AND RESULTS

4.1 Stage 1 Simulation

4.2 Stage 2 Simulation

4.3 Summary

## 5 CHAPTER 5: ANALYSIS AND DISCUSSION

5.1 Stage 1 Key Analysis – select query response times

5.2 Stage 1 Key – Pseudo Code Analysis

5.3 Stage 2 Key Analysis – Scalability Test

5.4 Development Directions

5.5 Summary

## 6 CHAPTER 6: CONCLUSIONS

BIBLIOGRAPHY

APPENDICES

Appendix A: The Abstracts of the paper
List of Figures

Figure 1. Digital Hospital (EZCARETECH, 2010) ................................................................. 19
Figure 2. 4-LESS(EZCARETECH, 2010) ............................................................................. 20
Figure 3. General Medical Information System (EZCARETECH, 2010) ............................. 21
Figure 4. Integrated Medical Information System (BIT, 2010) ............................................ 21
Figure 5. OCS(BIT, 2010) ..................................................................................................... 22
Figure 6. System Architecture and Environment (BIT, 2010) .............................................. 22
Figure 7. Program Summary (BIT, 2010) ............................................................................. 23
Figure 8. Image Migration Process (INFINITT healthcare co, 2010) ................................. 24
Figure 9. Architecture of storage and query of encrypted data (Lianzhong Liu, 2009 p.24) ....... 31
Figure 10. DAS model and architecture (Haocong et al., 2009) ........................................... 35
Figure 11. Self tuning process (Haocong et al., 2009) ............................................................ 36
Figure 12. Plain text process using Paillier encryption (Song & Park, 2009) ......................... 39
Figure 13. Integration of CLR inside SQL (Blakeley et al., 2008 p.1088) ......................... 42
Figure 14. SQL Server encryption key hierarchy with TDE and EKM (Microsoft, 2010a) .... 45
Figure 15. Encryption Functions (Oracle, 2010) ................................................................. 46
Figure 16. Structure of prototype EMR database ................................................................. 51
Figure 17. Detailed Schema of prototype EMR ................................................................. 53
Figure 18. Class Diagram for Random data generator, Performance checker and Application .... 59
Figure 19. Sequence Diagram of AES-DAS using Bucket Index and Bloom filter ............... 60
Figure 20. The Bloom Filter and the LINQ Usage ............................................................... 62
Figure 21. Sample data of Patient_Base ............................................................................... 66
Figure 22. Sample data of Patient_Identity .......................................................................... 66
Figure 23. SQL Profiler when LINQ is used ....................................................................... 67
Figure 24. Query Plans for Aggregation Query .................................................................... 67
Figure 25. Response Time of Select query test under 10K records ..................................... 77
Figure 26. Response Time of Select query test between 10K and 2M records ...................... 80
Figure 27. Logarithmic scale between record size and ch. type string ................................ 81
Figure 28. Performance Ratio among Bloom filter, HashSet, HashTable ............................. 82
Figure 29. Bulk insert test on EMR ..................................................................................... 83
Figure 30. Bulk insert scalability test .................................................................................. 84
Figure 31. String select query scalability test ...................................................................... 85
Figure 32. Numeric select query scalability test ................................................................. 86
Figure 33. Range select query scalability test ..................................................................... 88
Figure 34. Range Count Select query scalability test .......................................................... 90
Figure 35. Proposed Range select query scalability test ...................................................... 91
Figure 36. Aggregate query scalability test .......................................................................... 93
Figure 37. Current development and remaining work.........................................................94
List of Tables

Table 1. Three proposed database operational environments..................................................14
Table 2. List of Query Conditions..................................................................................................52
Table 3. List of experimental databases (Stage 1 Simulation).....................................................54
Table 4. List of experimental databases (Stage 2 Simulation)......................................................55
Table 5. The measurement issues according to the DB operation environment..........................57
Table 6. Simulation environment ................................................................................................58
Table 7. Pesudo Code for Range queries using Bucket Indexes .................................................61
Table 8. Query Plan Comparison for Aggregation Queries .........................................................68
Table 9. Response Time of Insertion Test (10K records)...............................................................69
Table 10. Response Times of Select Query Test (milli-seconds)....................................................70
Table 11. Bulk Insert result ........................................................................................................71
Table 12. Plain-text Scalability Simulation (milli-seconds)............................................................72
Table 13. MS built-in Scalability Simulation (milli-seconds).........................................................73
Table 14. AES-DAS using Bucket Index Scalability Simulation (milli-seconds)............................74
Table 15. Stage1 Simulation – Pseudo code & Analysis .................................................................79
List of Publications


The abstracts of the papers are in the Appendix A.
List of Abbreviations

AES: Advanced Encryption Standard
ASP: Active Server Page
API: Application Programming Interface
CLR: Common Language Runtime
CORBA: Common Object Request Broker Architecture
CPU: Central Processing Unit
CRUD: Create, Read, Update and Delete
DACS: Digital Access Connect System
DAS: Data as Service
DB: Database
DBMS: Database Management System
DBA: Database Administrator
DBO: Database Operator
DCOM: Distributed Component Object Model
DES: Data Encryption Standard
DSP: Database Service Providers
DSS: Decision Support System
DW: Data Warehouse

EHR: Electronic Health Record
EMR: Electronic Medical Record
EKM: Extensible Key Management

HIT: Health Information technology
HICT: Health Information and Communications Technology
HIS: Hospital Information System

JIT: just-in-time

LINQ: Language Integrated Query
Improving Trust and Securing Data Accessibility for e-health decision making by using Data Encryption Techniques

LIS: Laboratory Information System

MIBS: Medical Information Business System

MSIL: Microsoft Intermediate Language

MVC: Model-View-Controller

M-View: Materialised View

OCS: Order Communication System

OLAP: On Line Analytical Processing

OOP: Object Oriented Programming

ORM: Object to Relation Mapping

OS: Operating System

PACS: Picture Archiving Communication System

PH: Privacy Homomorphism

PHP: Hypertext Pre-processor

QBE: Query by Example

REST: Representational State Transfer

RDB: Relational Database

RDBMS: Relational Database Management System

RIS: Radiology Information System

RSA: Rivest, Shamir and Adleman

SOA: Service Oriented Architecture

SOAP: Simple Object Access Protocol

SQL: Structured Query Language

STBucket: Self Tuning Bucket

T-SQL: Transact-SQL

TDE: Transparent Data Encryption

WCF: Windows Communication Foundation

XML: Extensible Mark-up Language
Statement of Original Authorship

The work contained in this thesis has not been previously submitted to meet requirements for an award at this or any other higher education institution. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made.

Signature: __________________________

Date: __________________________
Acknowledgments

This thesis was written with supervisory assistance from Dr. Tony Sahama, Prof. Steve Kim and Dr. Adpro Renato Ianella. I am indebted for their support and grateful to have been able to work with such supervisors. Also, I am thankful to Prof. Ji Hong Kim for enabling me to be involved in researching database encryption schemes for the first time.

Thanks are also due to my family including my mother in Korea who gives me strength in daily life and happiness. They are always patient and waiting for my research journey to be well docked and the reason why I can continue my life in Australia. And I feel also thankful Kang Nam Moon who started this research course together. He gave me an insight while I was on the start of the research such as literature review part.

I believe God is always helping people who help themselves and this research could be done under God’s protection. This research outcome is not the end of my research goals but it is the start of my research journey.
Chapter 1: Introduction

This research study is to build a prototype EMR and to measure query response times under the three environments; plain-text, AES-DAS which is an outsourcing model, and Microsoft built-in encryption model which is an in-house model. Particularly, the range query response times are compared between a normal AES-DAS and AES-DAS using Bucket Index. Additionally, the character typed data encryption, the numeric typed data encryption, aggregate queries in the three environments are evaluated and tested fully scaled from 5K to 2560K records. In the case of the e-health arena, it aims at both patients’ private data and the stakeholders’ faster data accessibility through this project.

1.1 BACKGROUND

Contributions of Health Information technology (HIT) continue to have a significant impact on Patient Care, the e-health sector and Medical Decision Making processes.

E-health is the combination of electronic communication and information technology in support of decision-making processes for the health sector. The widespread adoption of e-health is vital in driving knowledge sharing, safety and quality in health care. However, health information and communications technology (HICT) alone will not dramatically improve care and reduce costs for health care services. Even when information is electronic, it is not always freely shared across organisational boundaries due to multiple constraints, barriers and problems with the culture of information sharing.

Unlike the promising e-health data sharing, illegal data leakage and data theft in the DAS (Data as Service) model could become a major obstacle to preserving patients’ privacy. According to news media in South Korea, cases have been reported of data theft by professional data service providers. The importance of managing data securely is of national concern. Under DAS models, there are three stakeholders: database owners, professional data service providers and clients. Out of all of the DAS models, it is the semi-DAS model which allows the database owners to be in charge of preserving the security of their data sets. In order to prevent the data thefts, the semi-DAS model is preferred to be utilised in healthcare platforms.

In order to guarantee faster result sets while using an encrypted database, three research areas are of concern. These are: ‘Order Preserving Encryption’, ‘Privacy Homomorphism’, and ‘Bucketisation’ (or Partition). These techniques will enable faster and more secure e-health decision making because they do not decrypt the whole of the encrypted data in the databases. Enhanced e-health security is the primary goal in e-health
data management models. However, there are some disadvantages when using the techniques above. The first one ‘Order Preserving Encryption’ is not as secure. The second one ‘Privacy Homomorphism’ is also proved to be insecure against a ciphertext-only attack. The third one ‘Bucketisation’ (or Partition) uses more additional storages and computation and its direct arithmetic is not possible. For these reasons, in this research project, the custom Bucket Index is utilised and designed. In this project, AES-DAS which means the DAS model encrypted by AES (Advanced Encryption Standard) is adopted and compare it with AES-DAS using Bucket Index and with an in-house model which is a DBMS (Database Management System) vender’s built-in encryption scheme. The AES-DAS using Bucket Index is assumed to be an ideal technique to be cost-effective because it is an outsourcing model and is much more secure than any plain-text simple databases.

1.2 CONTEXT

This study focuses on obtaining an outcome regarding the shortest SQL (Structured Query Language) response times under the three environments; plain-text, AES-DAS which is an outsourcing model and Microsoft built-in encryption which is an in-house model. The fundamental functionalities in databases are CRUD (Create, Read, Update and Delete) records. Due to the Read operations are heavily used than Create, Update or Delete, select query response time is a major concern because most of the query types are from the Read operation.

Table 1. Three proposed database operational environments

<table>
<thead>
<tr>
<th>Environment</th>
<th>Outsourcing Model</th>
<th>Encryption</th>
<th>Issue</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain-text</td>
<td>Yes or No</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>AES-DAS</td>
<td>Yes</td>
<td>Yes</td>
<td>Range query response times are different in the case of using Bucket Index as well</td>
</tr>
<tr>
<td>MS built-in Encryption</td>
<td>No</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>

1.3 PURPOSES

This study sets out to evaluate how the prototype EMR should be constructed to guarantee faster query response times even if the original data is encrypted. There are two main aims of the study. The first one is to evaluate the query performances including retrieving character type, numeric type, range values, range values using Bucket Index and aggregation type under the three proposed environments; Plain-text, custom AES-DAS and MS built-in Encryption scheme. The second one is to investigate if the first one’s results trend is consistent even though the databases become bigger from 5K to 2560K record-sized.

To investigate more effective health care decision-making when sharing sensitive and critical information based on the proposed data encryption technique which is the AES-DAS
using Bucket Index will be a progress for faster data accessibility and establishment of trust in e-health arena.

All the purposes are as follows:

- Evaluate the query performances including (string, numeric, range and aggregate) among Plain-text which is not encrypted at all, the AES-DAS using Bucket Index which incorporates currently well-known symmetric key encryption AES (Advanced Encryption Standard) and the Microsoft built-in RDBMS (Relational Database Management System) security in Microsoft SQL Server
- Evaluate the custom Bucket index engine using a Bloom filter saves query connection times and its utilisation
- Evaluate the trend according to the scalability from 5K to 2560K record-sized to see if there is any threshold for the consistent performance or it depends on the three simulation environments.

1.4 SIGNIFICANCE

This research includes two major simulations: The first one is to evaluate the query performances under the three environments. The second one is to evaluate the trend depending on the scalability. This research is an outstanding simulation scope for the encrypted database with performances. On top of this, further research can be carried out in some topics as the research under cloud computing or data-warehousing. Another significance is that this research will be a reference to some stakeholders who are preparing for an outsourced database maintenance model because in this research, many implementations and designs are considered into the outsourced database maintenance model – here this is used as the term ‘AES-DAS using Bucket Index’

If this research topic can be adopted in e-health arena which takes a high level of secured communication and preservation of the data, the history of all data for patients will exist for decades and will be used for finding solutions for diseases. The encryption method is a well-established technology for protecting sensitive data. However, once encrypted, the data can no longer be easily queried and sharing will be difficult for users. Nonetheless, such encrypted data can be shared between machines, and that is significant as far as information privacy is concerned. The performance of the database depends on how the sensitive data is encrypted and managed, and this is a key issue when it comes to producing a user-readable version. This study brings the following potential benefits over the current medical IT environments:
• AES-DAS model using Bucket Index will save range query response time even if the original data are encrypted
• AES-DAS model using Bucket Index will be a good outsourcing model for e-health decision makings
• The IT department among the e-health stake holders can have options to choose the encryption techniques properly based on the simulation results such as comparison between the custom AES-DAS and the MS built-in security.

1.5 THESIS OUTLINE

Chapter 2: Literature review introduces
  • Introduction to Medical Solutions such as EMR
  • Introduction to Current Cryptography Technologies
  • Bucket Index, Bloom filter and related issues
  • DAS model and Query Processor
  • Extensible Software Frameworks and Reviews

Chapter 3: Research design defines how to construct the simulation environments (Stage1 & Stage2) including a prototype EMR with Randomly Generated Dummy Data. It defines how to build up specific databases (Plain-text, AES-DAS using Bucket Index, MS built-in Encryption scheme) and check query performance measurement.

Chapter 4: The Stage 1 simulation result is the main outcome of the research. The result is categorised such as character type, numeric type, range query, range query using Bucket Index and aggregate query. It demonstrates graphs for easier analysis and some screen shots how the simulation goes. The Stage 2 simulation is based on the Stage 1 simulation. However, it focuses on the scalability from 5K to 2560K record-sized.

Chapter 5: The analysis with regard to the query response times (performances) according to the categories is explained by graphs and tables.

Chapter 6: Conclusion mentions the importance of database encryption techniques in the e-health arena and the focused result from the custom AES-DAS using Bucket Index and the scalability test when adopting an outsourcing model.
1.6 SUMMARY

In this chapter, the background of the reasons why we should keep the data securely in the e-health arena was identified. On the maintenance of the e-health data securely, there is a performance issue to support e-health decision makings. In this project, I suggest an outsourcing model which is a DAS (Data as Service) using AES (Advanced Encryption Standard) and Bucket Index (Partitioning). Its query performance is compared with Microsoft built-in encryption model and pure Plain-text model. This is the significance of this research in a matter of e-health database encryption technique and its scalability as well. This chapter outlines next chapters’ topics.
Chapter 2: Literature review

2.1 INTRODUCTION TO MEDICAL IT SOLUTIONS INCLUDING EMR AND EHR

2.1.1 Terminologies in medical IT solutions

The stakeholders such as researchers, doctors, nurses, patients and financial departments need many automated and electronic systems to support more accurate and detailed e-health decision making. Therefore, in this context, many e-health terminologies need to be issued. The e-health terminology review leads to understand the concepts that how this research is coupled into it. Korea has seemed to develop a lot of medical + IT solutions recently. Thus, this review will help to look into e-health arena.

EZCARETECH describes e-health terminology:

“DSS (Decision Support System): A system that analyses business data and supports information needed for making decision. It enables accurate decision making in a variety of situations.

DW (Data Warehouse): An integrated analysis system in which necessary data are obtained from separated systems and archived in a centralised repository, so that users can have access to them at any time.

EHR (Electronic Health Record): An extension of the Electronic Medical Record which aims at prevention of diseases and improvement of diagnosis and treatment by computerising not only clinical data of a patient but also all health-related records of an individual.

EMR (Electronic Medical Record): A computerised system for managing and research all patient’s medical records. An electronic version of medical record that offers accurate and complete health information and supports decision makings based on medical knowledge, replacing traditional paper charts.

HIS (Hospital Information System): A hospital’s core system that enables sharing of accurate and consistent data with other departments of a hospital through integration of hospital information and computerisation of work process. It consists of medical treatment information system, administration information system, medical treatment support system, business administration system, etc.

OCS (Order Communication System): A system that offers a DB (Database) in which a variety of medical information and patients’ data are stored and transfers a doctor’s prescription to the corresponding medical department through a communication network. It
Improving Trust and Securing Data Accessibility for e-health decision making by using Data Encryption Techniques

is an information system that manages all processes from patient registration to medical treatment to billing and allows follow-up of procedures and result. It is often confused with HIS.

PACS (Picture Archiving Communication System): A digital medical image archiving and transferring system that digitises the image obtained from a radioactive imaging device and transfers them along with the medical record to each terminal through a network.” (EZCARETECH, 2010)

The digital hospital concept by EZCARETECH is below.

Figure 1. Digital Hospital (EZCARETECH, 2010)

The digital hospital contains a lot of modules to support the work process and maintenance of the hospital. Here, EMR is a major concern for the research project. The data are all bound to each other and they can be utilized in B2B and B2C. The figures are utilised from the EZCARETECH web site.

2.1.2 Medical IT solution examples in South Korea

2.1.2.1 4LESS – ezCaretech

Medical Solutions require 4 major concepts which are ‘Chartless’, ‘Sleepless’, ‘Filmless’ and ‘Paperless’. These are key concepts that a medical solution needs to be directed toward. According to EZCARETECH (2010), it introduces a noticeable product named ‘4LESS’. 4LESS is an indispensable product of a digital hospital, enabling hospital staffs to focus on important matters, freed from drawing up, storing and managing paper documents. It also strengthens competitive power of the hospital through improvement of workflow and effective utilisation of precious resources. It emphasises ‘Chartless EMR’, ‘Sleepless HIS’, ‘Filmless PACS’ and ‘Paperless Groupware’.

The company focuses on two key areas for data mining that analyses medical treatment and its management system. The concept of the product named ‘4LESS’ is below.
Here, EMR needs to fulfil the environment of ‘Chartless’, ‘Slipless’, ‘Paperless’ and ‘Filmless’. In this project, the data is encrypted in a secured manner. And when the data comes from the data source, it needs to decrypt and be presented as human-readable. Therefore, the processing time to decrypt and the searching time to look into the whole database are important factors. In addition, they don’t need to be printed out, but the original data should be kept in a secured manner.

2.1.2.2 ezCare-DW

The data collected from medical solutions are extensive and to be managed securely. However, the data or the datasets can be utilised for researchers and marketers through data warehousing and data mining. According to EZCARETECH (2010), ezCare-DW offers a data warehouse system in which the data required by hospitals are critically analysed based on rich experiences in medical IT projects and outcomes from the experts. By enabling numerous data accumulated from past to present, ezCare-DW is the best medical DW allowing analysis and search of various types for medical research and management results. The general medical information system by EZCARETECH is below.
Breaking down all the tasks and projects under an integrated medical information system will satisfy to guide all the stakeholders what they should do and how to do the tasks and projects. According to BIT (BIT, 2010), Integrated Medical Information System provides all-inclusive and innovative tools to upgrade both office and clinical efficiency, in order to assist patient care. OCS, EMR, LIS (Laboratory Information System), RIS (Radiology Information System), and PACS provides on-the-spot and easier access to patient charts, laboratory results, previous orders, payment records, and a lot of other services that make smooth efficient communication among every department in the hospital. Not only that, it addresses all the critical aspects of managing a profitable hospital and enhances the hospital’s ability to provide cost-effective, distinctive care, yet easy to use applications with the latest technologies available.

The integrated medical information system by BIT is below.
2.1.2.4 OCS – BIT

Figure 5. OCS(BIT, 2010)

Figure 5 depicts an Order Communication System which takes many interactive messaging systems and considers many situations and exceptions under the OCS. It has administrative, treatment support, treatment and management information from on a daily basis and an archiving accomplishment system.

2.1.2.5 System Architecture and environment – BIT

The system architecture of OCS by BIT is below.

Figure 6. System Architecture and Environment(BIT, 2010)
The system above requires clustering of the OCS to prevent any hardware failure such as hard disks fail and power redundancy. It is a kind of Client-Server based solution but supports web based as well. However, most of the medical solutions nowadays require a data warehousing system to support better e-health decision makings.

2.1.2.6 Pharmacy Information Solution – BIT

According to BIT (BIT, 2010), there are several features in the pharmacy information solution of BIT seen below:

- It helps the patient to take medicine more precisely and increases the credibility of the pharmacy
- It can interface BIT medicine databases with the hospital OCS so that information such as pharmacology content, and medicine search engine can be available to users in real time
- It provides management system for all types of medicine in the inventory
- By calculation the statistical figure for products and sales volume, it enhances the owners of the pharmacy to overview the overall pharmacy management
- Users can receive various data regarding any type of medicine they wish to search

Figure 7. Program Summary (BIT, 2010)
2.1.2.7 DISWEB (Communicable Disease Surveillance Internet System) – BIT

Nowadays, the e-health system is evolving to a more open communication system. More and more interactive web based systems can alert residents to be aware of a certain infective disease. According to BIT (BIT, 2010), DISWEB web site provides every kind of disease related information such as remote report of disease outbreak from guard monitoring medical institution or educational institution, seasonal epidemic information for citizen, overseas contagious disease outbreak for tourists and analysis result of contagious disease database for experts.

2.1.2.8 PACS replacement – INFINITT

The PACS is one of the compulsory solutions in medical solutions. It has a functionality of ‘Chartless’ and ‘Filmless’. It enables clinicians to analyse their patients’ status and keep the data in a safe and an archived manner. According to INFINITT healthcare co (2010), it has several medical imaging products. However, it provides fast and accurate PACS replacement and migration services for transporting diagnostic images and patients’ electronic medical records in archives. The company’s knowledge and experience is a valuable asset through all phases of a successful migration and an innovation in a PACS healthcare.

The patients’ image migration process by INFINITT is below.

![Figure 8. Image Migration Process (INFINITT healthcare co, 2010)](image)
2.1.3 EMR requirements

As one of the most widely used in medical solutions to support e-health decision making, the EMR can be the most fundamental system among the medical solutions. According to Essin & Lincoln (1994), the EMRs should have certain requirements such as atomicity, authenticity, persistence, flexibility of representation and retrieval, semantic integrity, interoperability, process ability, performance and security. In order for the healthcare process to be most effective, the medical record must contain accurate and complete information that reveals the details of people’s lives and their medical histories. In the context of security, there are legal and ethical requirements that the records should be kept secure and confidential so that each individual’s privacy is preserved.

Because the EMR includes each individual’s privacy and history, it should incorporate with a large set of old archived data. The archived data could be preserved in many years.

2.1.4 Key management and related issues on privacy of EMRs

To make the EMR in a secured manner, there are some issues to affect the stakeholders and the environments. According to Benaloh, Chase, Horvitz & Lauter (2009), there are key management and related issues on privacy of EMRs.

- Key Revocation: a patient always has the option of changing symmetric or asymmetric keys
- Emergency response: patient might be given the option to wear or carry an enhanced medic-alert bracelet
- Patient Key Management: patient could keep a hardware device that stores a backup of their root secret key and security
- Doctor/Device Key Management: doctors could potentially have to store, manage, and protect local copies of a secret key for each their patients
- Usability: the system might be preset with several different options defining default hierarchies and sets of keys to issue to doctors, family members, devices, etc.

There are a lot of stakeholders related to EMR. Therefore, the facilities for each stakeholders needs to be satisfied and the access control for them needs to be managed properly.
2.1.5 Designing an EMR using design patterns

How to design an EMR is very complex and needs to consider many exceptions under the secured manner. To build up the system more effectively and efficiently, the EMR can adopt many frameworks. Frameworks include a lot of APIs (Application Programming Interface) and design patterns. In this research project, many design patterns can be considered to adopt. One of the most famous design patterns is the Singleton design pattern. It loads some information or functionalities only once statically in the initial stage, many other modules can invoke the Singleton design patterned module and share its information or its functionalities. According to Nisanbayev, Na, Lim & Ko(2009), there are several design patterns to construct a prototype of EMR.

The design patterns used are below:

- Singleton pattern: in the class model, a registrar class which serves as a central hub for all patient registration issues
- Abstract Factory pattern: an abstract factory pattern allows provides an interface for creating families of related objects without specifying their concrete classes
- Factory method pattern: the factory method pattern generally eliminates a need to bind application specific classes into codes
- Façade pattern: this useful pattern provides a good solution to an otherwise complicated issue in design
- Proxy pattern: this pattern is used to avoid creating an expensive object unless there is a real need for it

2.1.6 Electronic Health Record - a superset of Electronic Medical Record

According to Ebadollahi et al.(2006), Electronic Health Records and Electronic Medical Records have similar concepts. However, with regard to investigating the implementations of EHR and EMR they seem to be different in the usages and the concepts. EHR can be beyond EMR which are referred to. EHR aims at not only obtaining patient records across those separated systems, but also across geography and institutions. Furthermore, one of the e-health stakeholders, the government is considering EHR as below.

- Informed clinical practice
- Interconnection of clinics
- Improved population studies
Ebadollahi et al. (2006 p.1000) also mentions that EHR can be developed under the following nature: “is a”, “physically related to”, “spatially related to”, “temporally related to”, “functionally related to”, and “conceptually related to”.

2.2 INTRODUCTION TO CURRENT CRYPTOGRAPHY TECHNOLOGIES

This project particularly uses AES (Advanced Encryption Standard) which is a symmetric key encryption. However, there are some other encryption techniques that should be reviewed.

2.2.1 DES (Data Encryption Standard)

In this project, Both AES and DES can be evaluated. However, AES replaces the DES technology because it is more secure and more compliant to the software implementations.

2.2.1.1 Introduction to symmetric ciphers (cited in Smart, 2003 p.89)

A symmetric cipher works using the following two transformations

\[ c = e_k(m), \]
\[ m = d_k(c) \]

Where
- \( m \) is the plain text,
- \( e \) is the encryption function,
- \( d \) is the decryption function,
- \( k \) is the secret key,
- \( c \) is the cipher text

The DES cipher is a variant of the basic Feistel cipher. The outstanding property of a Feistel cipher is that the round function is invertible irrespective of the choice of the function in the box marked \( F \). To see this acknowledgement that each encryption round is given by

\[ L_i = R_{i-1}, \]
\[ R_i = L_{i-1} \oplus F(K_i, R_{i-1}) \]

Hence, the decryption can be performed via

\[ R_{i-1} = L_i, \]
\[ L_{i-1} = R_i \oplus F(K_i, L_i) \]

(Smart, 2003 p.96).

2.2.1.2 Overview of DES Operation
According to Smart (2003 p.98), basically DES is a Feistel cipher with 16 rounds. To sum up, the DES cipher operates on 64 bits of plain text in the following manner:

- Perform an initial permutation
- Split the blocks into left and right half
- Perform 16 rounds of identical operations
- Join the half blocks back together
- Perform a final permutation

The final permutation is the inverse of the initial permutation; this permits the same hardware/software to be used for encryption and decryption. The key schedule provides 16 round keys of 48 bits in length by selecting 48 bits from the 56-bit main key.

2.2.2 AES

2.2.2.1 Rijndael Algorithm

Rijndael is a name of the authors who proposed the AES specification. The AES was proposed to replace old DES technology in the U.S.

Rijndael uses a repeated number of rounds to obtain security and each round consists of substitutions and permutations, plus a key addition phase.

“Rijndael is a parameterised algorithm in that it can operate on block size of 128, 192, or 256 bits; it can also accept keys of size 128, 192, or 256 bits. For each combination of block and key size a different number of rounds is specified” (Smart, 2003 p.103).

“Rijndael operates on an internal four-by-four matrix of bytes, called the state matrix” (Smart, 2003 p.103).

\[
S = \begin{pmatrix}
S_{0,0} & S_{0,1} & S_{0,2} & S_{0,3} \\
S_{1,0} & S_{1,1} & S_{1,2} & S_{1,3} \\
S_{2,0} & S_{2,1} & S_{2,2} & S_{2,3} \\
S_{3,0} & S_{3,1} & S_{3,2} & S_{3,3}
\end{pmatrix}
\]

“which is usually held as a vector of four 32-bit words, each word representing a column. Each round key is also held as a four-by-four matrix” (Smart, 2003 p.103).

\[
K_i = \begin{pmatrix}
k_{i,0} & k_{i,1} & k_{i,2} & k_{i,3} \\
k_{i,0} & k_{i,1} & k_{i,2} & k_{i,3} \\
k_{i,0} & k_{i,1} & k_{i,2} & k_{i,3} \\
k_{i,0} & k_{i,1} & k_{i,2} & k_{i,3}
\end{pmatrix}
\]
2.2.2.2 AES High-level description of the algorithm

According to Schneier et al. (2000 p.4), there were 5 candidates to be adopted as AES (Advanced Encryption Standard). In their opinion, Twofish and Serpent have good safety factors. MARS is close, but RC6 and Rijndael clearly need more rounds.

“Simplest is the NIST criteria that’s hardest to describe. Lines of pseudo code, number of mathematical equations, density of lines in a block diagrams: these are all potential measures of simplicity” (Schneier et al., 2000 p.8). The reason why symmetric key is chosen compared to asymmetric key is that in the database, the symmetric key management is easier than asymmetric key management. In addition, the performance test can focus on the encrypted fields in tables rather than the structure of encryption.

2.2.3 Public key encryption - RSA (Rivest, Shamir and Adleman who first publicly described it)

2.2.3.1 Basic public key cryptography concept

Here the public key encryption is not a proper way to encrypt the database columns’ values. It is more suitable to data access control authentication. Smart (2003 p.143) mentions that the public key cryptography uses two identical keys, one public and one private for easier key management.

Message + Alice’s public key = Ciphertext,
Ciphertext + Alice’s private key = Message

2.2.3.2 RSA

Smart (2003 p.149) mentions that the RSA algorithm was the first public key encryption algorithm in the world, and it has endured the test period. For example, if Alice wishes to enable anyone to send her secret messages, she first chooses two large secret prime numbers $p$ and $q$.

Smart (2003 p.150) mentions,

Alice calculates $N = p \cdot q$

Also chooses an encryption component $e$ that satisfies

$$\text{gcd}(e, (p - 1)(q - 1)) = 1$$

It is normal to choose $e = 3, 17$ or 65537 which are prime numbers. Alice’s public key is the pair $(N, e)$, which she can publish in a public directory. To calculate the private key Alice applies the prolonged Euclidean algorithm to $e$ and $(p-1)(q-1)$ to get the decryption exponent $d$, which should meet the requirement,
$$e \cdot d \equiv 1 \pmod{p - 1(q - 1)}$$

Alice keeps secret her private key, which is the triple \((d, p, q)\).

Now if Bob wishes to encrypt a message to Alice, he first searches for Alice’s public key that stands for the message as a number \(m\) that is strictly less than the public modules \(N\).

The ciphertext is then created by raising the messages to the power of the public encryption advocate modulo the public modulus.

$$c = m^e \pmod{N}$$

On receiving \(c\), Alice can decrypt the ciphertext to recover the message by exponentiation by the private decryption advocate.

$$m = c^d \pmod{N}$$

The database encryption from AES-DAS model is based on symmetric key encryption like AES (Advanced Encryption Standard). It is hard to manage private and public key at the same time. Thus, the implementation scope is limited to AES.

2.3 BUCKET INDEX, BLOOM FILTER AND RELATED ISSUES

2.3.1 Bucket index

The Bucket Index has a meaning of the partitioning. Therefore, the more the partitioning, the more detailed retrieving query results could be produced. It is very important how to design the Bucket Index in a matter of the number of the partitioning, the size of the partitioning, and the relationship between the encrypted original data and the partitioned data.

The Bucket Index is identical for character type data. The construction of the index should follow two principles, firstly the index filters false records efficiently, and secondly it should be safe enough not to leak the true value.

Numeric data needs equations and range query with “between … and”. An index supporting all computations does not exist, thus, it is possible to create different types of indexes according to the data type and their purposes.

“The index tries to translate the character string into numeric data, on which the primary query will be processed to filter the records roughly. Only the rest records need to be decrypted, and it will save lots of time. So the cipher index can improve the query efficiency greatly” (Yong, Wei-xin & Xia-mu, 2007). The researcher ‘Yong et al’ are the bucket originators.

The bucket index speeds up query time because it stores key information that the data exist in the database through additional fields. If the data exists, the boundary of the data becomes smaller than the decryption of the whole data in the table.
“We add a bucket index field in a database which is corresponding to a cipher character data field. When there is a query over the encrypted field, we convert the SQL clause into another clause on the additional numeric field. The translated SQL clause filters the records roughly, and then gets fewer records which need to be decrypted” (Yong et al., 2007).

The Bucket Index is different from the concept of index in a database. It stores some key information or how the information is spitted into many partitions. Usually, in the case of range query, firstly the rough coarse query is executed, then within the range; the second query decrypts the proper ranged fields and search for the exact data.

The structure of bucket index using bloom filter is below.

The left box means either Application or Web browser. There are different procedures between insert and select queries. In case of insert query, the data is stored in the database through store module (with encryption). In case of select query, it executes the coarse querying on encrypted data and the querying at first, then, through the query result filter, it decrypts a small range of data and finally returns the query result.

The Direct arithmetic means that the query can get some meaningful values with only one query. In this bucketisation, it is impossible to get desired data within one query process.

In the bucketisation, the select queries need to have at least two steps. The first query returns a coarse querying and the second query includes a small range of decryption. The query performance is based on the ‘SELECT’ and ‘QUERYING RESULT’. All the insertion needs to have encryption process to store in the database.

Figure 9. Architecture of storage and query of encrypted data (Lianzhong Liu, 2009 p.24)
2.3.2 Bloom filter algorithm

A Bloom filter simply answers “yes” or “no” even though it holds the original data. If this filter can be used as a Singleton patterned instance in a Bucket Index engine, this will save a database connection time in the case of “no” answer. Because it says “no” when the Bucket Index engine does not have the data to be queried.

Zhong et al. (2008 p.355) mentions that Bloom filters are a compact set of representations that support set membership queries with small, one-sided error probabilities. Zhong et al. (2008 p.356) mentions that standard bloom filters only support element insertions and membership queries.

A Bloom filter is used to speed up answers in a key-value storage system. Its algorithm is widely adapted in DBMS (Database Management System) like Oracle 10g R2 and in many programming languages like C, Java and Python. The background of Bloom filters according to Zhong et al. (2008 p.356) is below.

A Bloom filter that represents an n-object set \( S = \{s_1, s_2, \ldots, s_n\} \) using an array of \( m \) bits, initially all set to 0. To insert each \( s_i \), one computes \( k \) different hash functions on \( s_i \), producing (up to) \( k \) hash values in \{1,2,\ldots,m\}, and then sets all the bits corresponding to the \( k \) hash values to 1. Thus, each \( S_i \) has different hash functions when they are set.

Assuming there is no dependency between set elements, here we will explain how standard Bloom filters determine the optimal number of hashes (\( k \)) used per object. If \( k \) independent, uniformly random hash functions are used to construct an \( m \)-bit bloom filter for an \( n \)-element set, then the probability that an arbitrary bit in the Bloom filter is set to 1 is:

\[
B = 1 - \left(1 - \frac{1}{m}\right)^{kn}
\]

Consequently, the false-positive probability of this Bloom filter is the probability for \( k \) hashed bits of the queried object being all 1’s, or:

\[
f(k) = B^k = \left(1 - \left(1 - \frac{1}{m}\right)^{kn}\right)^k = e^{k \ln \left(1 - \frac{1}{m}\right)}
\]

Given \( m \) and \( n \), \( f(k) \) is minimised when \( B = 1/2 \) (each bit in the Bloom filter has equal probability to be 0 or 1), or (for large \( m \)’s):

\[
k = \frac{m}{n} \cdot \ln 2
\]

In this case, the false-positive probability \( f \) is minimised to:

\[
\left(\frac{1}{2}\right)^k \approx 0.6185^m
\]

(Zhong et al., 2008 p.356).
2.3.3 Encrypted Character and Numerical Data

According to Zheng-Fei, Wei & Bai-Le (2005), as for character type data, fuzzy (e.g. like) matching is important whereas as for numeric data, range (e.g. <,>) matching is important. Thus, as for character type encryption, it adds a field which stores the characteristic values of character data. So, it can filter away most records not relative to the query condition by checking a field. As for numeric data, creating an additional B\textsuperscript{+} tree before encryption is proposed.

2.3.4 Privacy Preserving SQL Queries

2.3.4.1 Problems of existing techniques on encrypted databases

According to Hyun, Justin & Dong Hoon (2008), SQL queries on encrypted database have some limitations. Using ‘Order Preserving Encryption’ method is not secure. Using ‘PH’ (Privacy Homomorphism) is also proved to be insecure against a ciphertext-only attack. Ciphertext-only attack means its encrypted fields in a database, an attacker or an untrustworthy DBO (Database Operator) can possibly see the leak or the encrypted data can be assumed. However, PH is useful for numeric encryption. Using ‘Bucketisation’ (or Partition) method also needs more storage and computation in databases and direct arithmetic is impossible. It should have partitioning fields that contains the key information indicating the data is in the DB or not.

2.3.4.2 Privacy Requirements

Hyun, Justin & Dong Hoon (2008) mentions a diversity of privacy enhancing security aspects such as the following.

- Anonymity. Anonymity ensures that a user may use a resource or service without disclosing the user’s identity.
- Pseudonymity. Pseudonymity can protect the user’s identity in cases where anonymity cannot be provided, e.g. if the user has to be held accountable for his activities.
- Unobservability. It ensures that a user may use a resource or service without others being able to observe that the resource or service is being used. Furthermore, it has to be prevented that an attacker can link various information about a user.
- Unlinkability. It ensures that a user may make use of resources and services without others being able to link these uses together.
2.4 DAS MODEL AND QUERY PROCESSOR

2.4.1 DAS

“Database-as-a-Service model is a new data management model that allows users to outsource their data to database service providers (DSP). Since data is stored in cryptograph form at DSP, the query efficiency becomes a critical problem. Existing solutions for this problem concentrate mostly on cryptograph index technology” (Wei, Feng, Danfeng & Guohua, 2010 p.688) Therefore, DAS can be a good outsourcing model. There are advantages and disadvantages from using a DAS.

The following advantages are:

- Reduced maintenance cost for a database
- Core and subsidiary tasks can be separated according to roles
- Keeping enhanced security even though the locations of databases are outside the domain

The disadvantages are:

- It should be clear on the locations of application servers as well
- Untrustworthy DBO can possibly hack the system
- Not secured compared to in-house model

“DAS models can be classified into three classes by the level of trust in DSP (Data Service Provider). The first class are Complete Trust DAS models that mean that the DSP is completely believable. The specific division is:

1. Owners, update the data periodically or occasionally to DSP
2. Database Service Providers, where the owner’s data are encrypted, they maintain the encrypted database and its security, and provide the users (including the appointed owners and ordinary users) with general operations in this database.
3. Client, submit queries about the owner’s data to DSP and get back results through internet” (Wei et al., 2010 p.688).

Wei et al. (2010) mentions that outsourcing a database to a third party aims at decreasing the cost of maintenance of DBMS. And bucket indexing will help the client to reduce the cost of a query without the decryption of the data.

2.4.2 Semi-Trust DAS

According to Danfeng et al. (2009), the second class Semi-Trust DAS model is more popular in theoretic research and practice. In addition, its Metadata for index could be stored in two kinds of types: firstly, metadata for index can be organised as relation structure and be manipulated using SQL; secondly, metadata can be stored in text files. However, it is
uncertain whether attacks occur in the provider. So the general strategy for data owners is to encrypt private information before transmitting them to database service providers.

2.4.3 Order preserving encryption
According to Jieping & Xiaoyong (Jieping & Xiaoyong, 2008), if the adversary gets to know plain text, he can construct the relationships between ordered plain text and ordered cipher text. In addition, the order preserving encryption has some limitations on the encryption technique and key management.

“Order preserving encryption is not originally proposed in a DAS model because it assumes that the server is trusted. However, it can be adapted to a DAS model easily through delegating all encryption related work to the client. Order preserving encryption means that ciphertext must keep the same order as plaintext. So an index built directly on ciphertext could be used for range queries. The advantage is that no false positive and additional work to the implementation of DBMS is incurred. Here false positive denotes the tuple which does not belong to the result set but returned from the server. The disadvantage is it can only be used for scenarios where the adversary has access to all (but only) ciphertext (the so called ciphertext only attack)” (Jieping & Xiaoyong, 2008).

2.4.4 Self Tuning bucket index
“By gathering and analysing query feedback, STBucket (Self Tuning Bucket) achieves adaptation to workload through online bucket splitting and merging. Experimental results show that STBucket is workload aware and performs well with reasonable overhead” (Haocong, Xiaoyong, Jieping & Pingping, 2009).

The self tuning bucket index architecture in DAS is below.

Figure 10. DAS model and architecture (Haocong et al., 2009)
As noted in Figure 10, the DAS model and its architecture could be used in AES-DAS using Bucket Index, the Query translator can be a Bucket Index Engine. It manages the temporary data which holds coarse data from the encrypted database. It may have partitioned (Bucket Index) information or the coarse encrypted data. It will search key information to retrieve real data inside the coarse data. If it has the key information, the Result filter will decrypt the coarse data and return the original data or any forms of manipulated data.

“Our approach is complementarities to current bucket based methods. The client can decide whether to use this self-tuning scheme on certain attributes. If the data distribution and workload behaviour on one attribute is known previously and hardly changed, a well designed bucket based method can be used to construct a bucket set. Otherwise the client can use our method to construct an initial bucket set, and decide whether to refine the bucket set periodically” (Haocong et al., 2009 p.103).

The self tuning process of bucket index is below.

Figure 11. Self tuning process(Haocong et al., 2009)

2.4.5 Data warehouse schemas

2.4.5.1 Star join schema
Star join schema is the basic style schema among data warehouse schemas. It has possibly one fact table and any number of dimension tables. Dimension tables have one simple primary key, whereas a fact table has a set of foreign keys that build up one composite primary key including related dimension keys.

2.4.5.2 Snowflake schema
Snowflake schema has an entity relationship looking like a snowflake. The main structure of the snowflake schema is similar to that of star schema; however, the snowflake schema’s dimensions are normalised whereas the star schema’s dimensions are not normalised. In the
real world, it is not heavily used as much as the star schema in data warehousing because the snowflake schema has multiple star join schemas.

The reason why data warehousing schemes are relevant to this project is that in many clinical workplaces, the data sources are not all compatible and integrated properly. So the system designer needs to understand the effectiveness of data warehousing and consider the heterogeneous environments. Data warehousing scheme is an important factor because currently the EMR or EHR can be constructed in heterogeneous systems in accordance with each department in a hospital. Thus, It is important to merge and separate the interest of the outcome in a secured manner.

2.4.6 Microsoft SQL Server 2008’s query processor

While Microsoft SQL Server 2008 was being developed and evolved, it actually used customer workloads to quantify performance improvement and to limit regressions. Grabs, Herbert & Zhang (2008) mentions that cost based query optimisation is of a statistical nature; query processing heavily relies on carefully tuned heuristics in the product. A common workload on a star schema generally includes a variable mix of ad-hoc and parameterised queries. In the cost and cardinality estimation, the query processor needed to provide added value and performance without the extra overhead of requiring database administrators to configure the feature manually. From the customer’s perspective, certain queries in their workload may be mere “important” than others. Over time, this small regression may outweigh the benefit gained from the less frequently run query. The model based testing of extending SQL Server’s Query processor has two key areas. One is schema modelling that is concerned with

- Number and Classification of Tables
- Cardinality
- Relationships
- Distribution of Data
  - “Skewed distributions tend to be very important in query processor testing in order to validate many of the algorithmic assumptions that are often made with respect to normalised or random data” (Grabs et al., 2008).

The other is query modelling that is concerned with

- Number of fact tables involved
- Number of dimension tables involved
- Predications on the dimension table
- Aggregation functions on fact tables
 Nested sub queries

 Expected selectivity

 - “The star join optimisations are designed to handle queries that have high selectivity very differently from queries that are non-selective
  
 To quantify query performance, the geometric mean is used rather than the arithmetic mean because it gives short and long running queries the same weight” (Grabs et al., 2008).

 As a further study, a multi-dimensional testing approach is necessary.

### 2.4.7 Semantic queries in databases

#### 2.4.7.1 Introduction

Lim, Wang & Wang (2009, p.1505) mentions that clinicians recording the diagnosis of a patient visit may use different disease codes for the same symptoms that the patient is exhibiting. One clinician might describe a patient diagnosis using the code for “Tumor of the Uvea”, while another might use the code for “Iris Neoplasm”. Thus, in order to obtain reasonable results from querying an EMR database, the processing system to query needs to understand the semantics of the query and the data.

#### 2.4.7.2 QBE (Query by Example)

According to Lim, Wang & Wang (2009, p.1508), QBE is a well-known concept in the database community. It was introduced in mid 1970’s and it is quite different from SQL in that it is a graphical query language. In the commercial products, QBE is widely used as graphical front-end for RDBMSs. There are two common characteristics for all the previous QBE work: the examples are used to specify a query that will be generated, the generated query is a normal query in terms that all the query conditions are defined on the base attributes in the underlying tables. The semantic QBE is very different from the traditional QBE problem. The real query associated with the users’ intention which is specified by the input examples is really hard to capture by a traditional SQL query.
2.5 NUMBER THEORY

2.5.1 homomorphic encryption

Number theory is concerned with the properties of numbers in general. Homomorphic encryption is a set of encryption in which a certain algebraic operation is executed on the plaintext and a possibly different algebraic operation is executed on the encrypted text. This encryption is one of the database encryption techniques but in the simulation scope, this is not included because Bucket Index techniques are used in this research project.

2.5.2 Generalised Paillier

![Diagram of Paillier encryption](image)

Figure 12. Plain text process using Paillier encryption (Song & Park, 2009)

Song & Park describes Generalised Paillier homomorphic encryption. The scenario of Generalised Paillier encryption usage is below.

1. Suppose there are three encrypted blocks
2. Calculate using Generalised Paillier encryption
3. Decrypt using the decryption key
4. The sum of plain text is the same as the product of the encrypted

The symbols and the method is below.

- \(m\) : the plaintext block
- \(m_{ij}\) : the value of the block row \(i \ (1 \leq i \leq n)\), and column \(j \ (1 \leq j \leq k)\)
- \(C_i\) : the encrypted row \((1 \leq i \leq n)\)
- \(K\) : the decryption symmetric key
$S_i$: the bit array of the result

Input: $C_i$, $K$

Output: $\sum_{i,j} m_{ij}$

In detail,

1. Using Generalised Paillier encryption, calculate like $C = \prod_{i=1}^{n} C_i$
2. Decrypt $C$ using the symmetric key $K$ $S = dec(K,C)$
3. Divide the decrypted $S$ by the same bit length
   At this point, the coefficient of $S$ is $k$
   $S = S_1 \circ S_2 \circ ... \circ S_k$
4. Print $\sum_{i=1}^{k} S_i$

(Song & Park, 2009).

### 2.6 EXTENSIBLE SOFTWARE FRAMEWORKS FOR EMRS

The extensible software frameworks for EMRs are techniques of how to build the performance checking application, how to use less codes and well organized structures. In this project, LINQ is incorporated into the performance checker for Bucket Index.

Extensive software frameworks deliver efficient and flexible maintenance of EMR. It will help to keep the maintenance cost and better scalability.

#### 2.6.1 LINQ (Language Integrated Query)

For the research project, this is an additional feature for the implementations. LINQ is basically providing an ORM (Object-Relation Mapping) to help the CRUD operation is well to be resided in the application code sides. As a custom AES-DAS model, this approach is an advantage due to the encryption / decryption logic that can be layered in the application side which can be an outsourcing model.

At this time, LINQ is only a feature that supports SQL Server. However, in the next version of .NET framework, it will support other databases like Oracle or MySQL.

To use LINQ, it requires adding System.Data.Linq.dll assembly and a namespace ‘System.Data.Linq’

Joseph C. Rattz (2007) mentions that
In LINQ, there are several components to help the development be easier and powerful.

- LINQ to Objects
- LINQ to XML (Extensible Mark-up Language)
- LINQ to DataSet
- LINQ to SQL
- LINQ to Entities

LINQ is not only just an easier way of querying but also a data iteration engine. The components related to DataSet, SQL and Entities are concerned with the research. In DataSet component, several DataRow Set Operators are introduced.

- Distinct
- Except
- Intersect
- Union
- SequenceEqual

Additional consideration is how to deal with typed datasets that are strongly connected to the schema of the database.

The trend from this concept requires ORM (Object to Relation Mapping) that it is a programming technique for converting data between RDB (Relational Database) and OOP (Object Oriented Programming).

This part was utilised into the journal [3] Dong-Il Shin, Tony Sahama, Jung-Tae Kim and Ji-Hong Kim (2010). “Issues on medical numeric data by using Encryption Techniques”, Proceeding of International conference of KIMICS, June 30-Jul 1, Gold Coast, QLD, Australia in Appendix B: The Abstracts of the publications.

2.6.2 Hibernate

JBoss community (2010) says “Hibernate facilitated the storage and retrieval of Java domain objects via object/relation mapping”.

Hibernate is an open source ORM framework for Java technology. It has similar features of LINQ, thus, it has Hibernate Query Language that is the basis of Java Persistence Query Language. Hibernate also provides data query and retrieval functionalities. Hibernate generates the SQL calls and helps the developer from handling manual result set and keeping the application portable to all RDBMS.

2.6.3 iBATIS

Apache software foundation (2010) says “The iBATIS Data Mapper framework makes it easier to use a relational database with Java or .NET applications. iBATIS couples objects
with stored procedures or SQL statements using a XML descriptor. Simplicity is the biggest advantage of the iBATIS Data Mapper over object relational mapping tools”.

iBATIS is a framework, which makes the mapping easier between a RDBMS and objects in Java, .NET, and Ruby on Rails. In Java, by using XML configuration files, the mappings are decoupled from the application logic and the SQL statements are embedded in the files. The result is a reduction in the quantity of code that a developer needs to connect to a RDBMS and the framework also supports closing the connections with RDBMS automatically.

2.6.4 .NET Database Programmability and Extensibility in Microsoft SQL Server

![Figure 13. Integration of CLR inside SQL](Blakeley et al., 2008 p.1088)

Blakeley et al.(2008 p.1088) mentions that managed code is MSIL (Microsoft Intermediate Language) executed in the CLR (Common Language Runtime) rather than directly by the operating system. Managed code applications obtain CLR services such as automatic garbage collection, runtime type checking, and security support. These services help provide a certain unilateral platform- and language-independent behaviour of managed-code applications. At run time, a just-in-time (JIT) compiler translates the MSIL into native code that can be run on Intel or compatible CPU (Central Processing Unit). During this translation, code should pass a verification process that checks the MSIL and metadata to know whether the code can be determined to be type safe.

SQL Server and the CLR have different internal models for managing system resources including memory management, synchronisation primitives, and thread scheduling. SQL Server interacts as the operating system for the CLR when it is hosted inside the SQL Server process (see Figure 13). SQL Server encapsulates all operating system primitives in a component called the SQL OS (Operating System). The Hosting Layer allocates assembly loading, security management, application domain management, and escalation policy when severe exceptions occur.
2.6.5 N-tier architecture

Usually N-tier architecture is a 3-tier architecture that is composed of presentation layer, logic (business) layer and data layer. It is important to allocate the application modules into the proper layers according to the usage of the modules that are presentation layer, business layer and data layer. This methodology will bring the application be maintained easily and extensible. According to these layers, the layers will be served in multiple different hardware that will be more secured, fault-tolerant and distribution of heavy traffic.

2.6.6 MVC (Model-View-Controller)

These days, the software architecture is fairly bound to web based architectures. The reason is that web applications follow the structure of traditional applications like windows application as they become complex. As the web-based architectures are being evolved, MVC pattern is one of the most popular design in building not only web applications but also normal applications. Particularly, the model is the domain-specific representation of data that the application operates. Many applications use databases for storing their states and information. In some cases, the model knows how to manage data access. There are many frameworks to implement MVC pattern by many programming languages. Some popular languages like Java, C/C++, C# and PHP (Hypertext Preprocessor) have a lot of MVC frameworks. The names of the frameworks are below and for web applications.

- Java: Spring MVC framework
- C/C++: Wt- Web tool kit
- C#, .NET Framework: ASP (Active Server Page).NET MVC
- PHP: CakePHP

2.6.7 CLR on MS SQL Server

According to Microsoft(2010c), unlike stored procedures, triggers and other types of code modules that can be exposed within SQL Server, a given SQLCLR routine is not directly to a database, but rather to an assembly catalogued within the database. Unlike their T-SQL (Transact-SQL) equivalents, SQLCLR modules get their first security restrictions not via grants, but rather than the same time their assemblies are catalogued. The CREATE ASSEMBLY statement allows DBA (Database Administrator) or database developer to specify one of three security and reliability permissions that dictate what the code in the assembly is allowed to do.
2.6.8 SOA (Service Oriented Architecture)

Service Oriented Architecture provides a way to remove problems concerned with the integration of heterogeneous applications in a distributed environment. The advantage of the SOA is to enable the reuse of the services and loosely coupling among the applications. The Java owned by Oracle and the .NET owned by Microsoft are major heterogeneous virtual machine frameworks.

A system architect can design SOA using a wide range of technologies, including:

- SOAP (Simple Object Access Protocol)
- REST (Representational State Transfer)
- DCOM (Distributed Component Object Model)
- CORBA (Common Object Request Broker Architecture)
- WCF (Windows Communication Foundation)

2.7 INTRODUCTION TO BUILT-IN DBMS SECURITY

2.7.1 Built-in MS SQL security and encryption technologies

Microsoft provides a combination of the encryption / decryption programming approach from a DBA (Database Administrator)’s point of view. Firstly, the DBA can decide the Master key of the database. Secondly, he can choose a certificate or an asymmetric key. Thirdly, he can choose a symmetric key which is a Triple-DES as default. Therefore, it is really difficult to guess the original data and to decrypt by an untrustworthy DBO (Database Operator).

2.7.1.1 Encryption Hierarchy

Microsoft (2010a) mentions that SQL Server encrypts data with a hierarchical encryption and key management infrastructure. Each layer encrypts the layer below it by using a combination of certificates, asymmetric keys, and symmetric keys. Asymmetric keys and symmetric keys can be stored outside of SQL Server in an Extensible Key Management (EKM) module.
Figure 14. SQL Server encryption key hierarchy with TDE and EKM (Microsoft, 2010a)

2.7.1.2 Encryption Mechanisms

Microsoft (2010b) mentions that SQL Server provides the following mechanisms for encryption:

1. Transact-SQL functions: individual items can be encrypted as they are inserted or updated using Transact-SQL functions. As an example of this is ENCRYPTBYPASSPHRASE and DECRYPTBYPASSPHRASE.

2. Asymmetric keys

3. Symmetric keys

4. Certificates: Digitally signed statement that binds the value of a public key to the identity of the person, device, or service that has the corresponding private key. The SQL Server creates the self-signed certificates following X.509.

5. Transparent Data Encryption: TDE is a special case of encryption using a symmetric key. TDE encrypts an entire database using that symmetric key. The database encryption key is protected by other keys or certificates, which are protected either by the database master key or by an asymmetric key stored in an EKM module.
2.7.2 Built-in Oracle 10g encryption technologies

Hall (2010) says “The Transparent Data Encryption (TDE) feature was introduced in Oracle 10g Database Release 2 to simplify the encryption of data within data files, preventing access to it from the operating system. Tablespace encryption extends this technology, allowing encryption of the entire contents of a tablespace, rather than having to configure encryption on a column-by-column basis.

2.7.3 Built-in MySQL encryption technologies

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES_DECRYPT()</td>
<td>Decrypt using AES</td>
</tr>
<tr>
<td>AES_ENCRYPT()</td>
<td>Encrypt using AES</td>
</tr>
<tr>
<td>COMPRESS(32.3)</td>
<td>Return result as a binary string</td>
</tr>
<tr>
<td>DECRYPT()</td>
<td>Decodes a string encrypted using ENCODE()</td>
</tr>
<tr>
<td>DES_DECRYPT()</td>
<td>Decrypt a string</td>
</tr>
<tr>
<td>DES_ENCRYPT()</td>
<td>Encrypt a string</td>
</tr>
<tr>
<td>ENCRYPT()</td>
<td>Encrypt a string</td>
</tr>
<tr>
<td>ENCRYPT(32.3)</td>
<td>Encrypt a string</td>
</tr>
<tr>
<td>IDAE()</td>
<td>Calculate MD5 checksum</td>
</tr>
<tr>
<td>OLD_PASSWORD(32.3)</td>
<td>Return the value of the old (pre-4.1) implementation of PASSWORD</td>
</tr>
<tr>
<td>PASSWORD()</td>
<td>Calculate and return a password string</td>
</tr>
<tr>
<td>SHA1(32.3)</td>
<td>Calculate an SHA-1 160-bit checksum</td>
</tr>
<tr>
<td>UNCOMPRESS(32.3)</td>
<td>Uncompress a string compressed</td>
</tr>
<tr>
<td>UNCOMPRESSED_LENGTH()</td>
<td>Return the length of a string before compression</td>
</tr>
</tbody>
</table>

Figure 15. Encryption Functions (Oracle, 2010)

In MySQL, it does not support built-in encryption architecture, but it supports various encryption functions.

2.8 REVIEW OF AGGREGATE-BASED QUERY

2.8.1 Aggregate-based Query in a Parallel Data Warehouse Server

Albrecht & Sporer (1999) mentions that a direct method to speed up queries is the use of parallel hardware. However, a different method specific to data warehousing is to statically preaggregate some of the results in order to avoid scanning the base relations.

In parallel aggregate query processing, each of the fragments can locally be aggregated to the first request level, and the final result can be constructed at the aggregated fragments at a collector level. The microeconomic cost model can offer several advantages over traditional ways:
- Local and less complex algorithms
- Integrated cost system for all resources
- Self regulating stability
- High overall performance

The consideration of pre-computing summary data and storing them as materialised views became popular with the emergence of data warehousing and the need for effective read-only access to aggregates in a multi-dimensional context. However, dynamic or adaptive strategies can try to optimise the set of pre-computed aggregates.

### 2.8.2 Partial Preaggregation

There are 3 types of joins:

- **Nested loop join**: its performance depends on the first accessed table’s processing scope. It is called as ‘Driving table’, thus, narrower the driving table, faster performance.
- **Sort merge join**: it is usually executed on a batch processing. Firstly after sorting each table, then execute merge join on them.
- **Hash join**: it uses more memory and CPU resources and needs equal join and reads joining tables only one time.

Larson (2002) mentions that preaggregating before a join is the most clear application of partial preaggregation. Partial preaggregation is a simple data reduction operator that can be applied to aggregation queries or queries with duplicated elimination. However, when the reduction is low, performance is not always good compared with performing no preaggregation or performing exact preaggregation.

### 2.8.4 Overcoming Limitations of Sampling for Aggregation Queries

Chaudhuri, Das, Datar, Motwani & Narasayya (2001) mention that decision support applications such as OLAP (On Line Analytical Processing) and data mining tools for analysing large databases are earning popularity. OLAP servers that answer queries involving aggregation can potentially gain from the ability to use sampling. However, there are doubtable factors such as data skew and low selectivity of queries on uniform random samples which are used for approximating results of aggregation queries over RDBMS.
There is a concept of M-View (Materialised View) that has a snapshot of a certain OLAP and guarantees faster output performance. And it may be physically implemented as a heap or a B tree index.

2.9 IMPLICATIONS

Based on the literature review above, in this research project, the Bucket Index and Bloom filter techniques are utilised to construct a prototype EMR. In addition, these techniques are compared with the Microsoft built-in encryption technique. It is important which CRUD (Create, Read, Update and Delete) operations on databases would be focused on. As an example, if the read operation in the Bucket index takes less response time than that of the MS (Microsoft) built-in encryption technique, the update, remove or create operations in the Bucket index would take less response time than those of the MS built-in encryption technique as well.

However, there would be some questions that are testable through empirical research.

- What kind of design pattern and collection classes can be implemented to construct the proposed custom encryption technique?
- Between custom encryption (AES-DAS using Bucket index) and MS built-in security in RDBMS, which one provides the fastest CRUD operation?
- How will the performance trend be changed as long as the databases become in a large scale?
- Is the custom AES-DAS using Bucket Index recommended to every CRUD operations compared to the MS built-in encryption technique?

2.10 SUMMARY

In this chapter, the medical terms and solutions such as EMR were introduced firstly. To adopt more secured EMR, the current cryptography technologies such as AES was introduced. Then, for an outsourcing model, Introducing Bucket Index which means database partitioning and Bloom filter which can be utilised in a Singleton design patterned as a part of Bucket Index engine. Also, other database techniques such as Homomorphism and Privacy Preserving briefly were mentioned. The Microsoft built-in encryption technique which can be used as in-house model was introduced and compared with the outsourcing model (AES-DAS using Bucket Index). In the project, the AES-DAS using Bucket Index will be compared with the MS built-in encryption technique and be mentioned the benefits of the technique.
Chapter 3: Research design

3.1 INTRODUCTION

The main aim of the research is to evaluate a proposed custom AES-DAS using Bucket Index comparing with other environments such as Plain-text and MS built-in encryption technique.

There are 2 simulation preparations to be carried out. The first one is to evaluate the query performances including retrieving character type, numeric type, range values, range values using Bucket Index and aggregation type under the three proposed environments: Plain-text, custom AES-DAS and MS built-in Encryption scheme. The second one is to investigate the first one’s results trend is consistent even though the databases become bigger from 5K to 2560K record-sized.

This research includes building a prototype of medical database schemas with a few proposed encryption techniques such as AES-DAS using Bucket Index and MS built-in encryption scheme. Additionally, the complexity in the DB schemas and their sizes will be a matter of concern to measure their query response times in a matter of CRUD operation.

Under the three database environments, there are modules needed in this project.
The implementation outcomes are,

- Random Data Generator
- Performance Checker
- Bucket Index Application including the Bloom filter
- A set of SQLs to measure the response times when using MS built-in encryption technique

Subsequently, there are six query categories to be compared with:

- Character typed data query (Read operation – Select query)
- Numeric typed data query (Read operation – Select query)
- Range query (Read operation – Select query)
- Range query using Bucket Index
  (Read operation, only applied into custom AES-DAS model)
- Aggregate query (Read operation – Select query)
- Total time to build up each database (Create operation – Insert query)
The reason why character typed data and numeric typed data are dealt in query performances is that they are the basic information to save in the database. Aggregate and range queries are heavily utilised to get some statistical data. The Delete function is not a compulsory because if the user wants to delete them, just delete query does not have to decrypt at all.

The Read operation is the most heavily used operation in DBMS. Therefore, in this project, I prepare for an application which measures particularly the select query response times.

### 3.2 METHODOLOGY

Two simulation stages are to be performed with regard to the encryption techniques and the scalability. The version of SQL is SQL:2008 is used and the DBMS is Microsoft SQL Server 2008

#### 3.2.1 Stage 1 Simulation Methodology

The stage 1 simulation includes 6 databases as following.

- Plain-text (10K records)
- MS built-in encryption (10K records)
- AES-DAS (10K records)
- AES-DAS using Semi-Bucket (10K records)
- AES-DAS using Bucket Index plus Bloom filter (10K records)
- AES-DAS using Semi-Bucket (2M records)

AES-DAS using semi-bucket is an intermediate outcome of the implementation in the project. AES-DAS using semi-bucket is not as secure as AES-DAS using Bucket Index.

The Performance Checker measures query response times among the different database maintenance models and the different predicates for queries. The Bucket Index application incorporates a Singleton-designed patterned memory that is the Bloom filter containing pre-partitioned field information. In the stage 1 simulation, if non-matched partitioning field information is queried in the Bucket Index application, it means there is no data matched in the DBMS. Therefore, the query stops searching and finishes. The detailed sequence diagram for the simulation flow is depicted in Figure 21 in chapter 4.

The structure of the EMR database is shown in Figure 16.
Figure 16. Structure of prototype EMR database

The table named ‘Patient_Record’ has the most relationships with other tables. In a real world application, the patient treatment records can be encrypted, however, the simulation does not include the treatment records as a test target because its type is a long text or binary type which can definitely erode its performance.

In the stage1 simulation, particularly, BirthDate field in the table ‘Patient_Identity’ is used for the range query between Plain-text, MS built-in encryption and Bucket Index including Bloom filter. In addition, Blood type field in the table is also tested for aggregate query between those models. Additionally, the performance comparison between alphabetical data and numeric data is dealt when they are encrypted.

The encrypted fields are distributed in 3 major tables below as the items in the simulation.

- Patient_Base Table
  - Address (nvarchar – alphabetical, character typed data)
  - MedicareNo (nvarchar – numeric typed data)
- Patient_Identity Table
  - BirthDate (datetime – has Bucket Index and Bloom filter values)
  - BloodType (nvarchar - alphabetical) : only used in aggregated query
- Patient_Record Table
  - Treat_gp (nvarchar - alphabetical data) : in this project, this field is not included in the simulation

Summing up the predicates of the queries is below.
Table 2. List of Query Conditions

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Content Type</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address</td>
<td>nvarchar</td>
<td>Character</td>
<td></td>
</tr>
<tr>
<td>MedicareNo</td>
<td>nvarchar</td>
<td>Numeric</td>
<td></td>
</tr>
<tr>
<td>BirthDate</td>
<td>datetime</td>
<td>Date and Time</td>
<td>Related to Range Query and Bucket Index</td>
</tr>
<tr>
<td>BloodType</td>
<td>nvarchar</td>
<td>Character</td>
<td>Related to Aggregate Query</td>
</tr>
</tbody>
</table>

The original data is composed of character typed, numeric typed, date time typed. Those are the fundamental value-type in databases. It is easy to check the birth date so as to calculate the specific ranged data when comparing with the Bucket Index performances.

To build up the encrypted database, among CRUD functions, C (Create) – insert queries are executed. After Creation, the R (Read) – select queries are performed. However, in this project, D (Delete) function is not compulsory because the records can be deleted by primary key or it does not necessarily have to be decrypted.

The number of records in the stage 1 simulation is 10K records each. In the stage 2 simulation, the scalable tests are performed from 5K to 2560K record-sized.

The Aggregate Query is to count the users grouped by same blood type such as ‘SELECT BloodType, COUNT(*) FROM Patient_Identity GROUP BY BloodType’

The detailed EMR is shown in Figure 17.
The procedure for the data management in this schema is presented as follows:

- The admin role user creates a patient base information.
- The admin role user maps the patient to GP.
- The GP role user creates patient identity information which is related to the patient base information, and also maps the information to SP when the illness is serious.
- The SP role user adds patient treatment records with the GP role user. The patient_record table is joined with the disease table and the patient_chart table.
that includes binary information such as images and further information and schedules.

- The Nurse Role users can only retrieve the data inserted by GP or SPs. However, the nurse can get permission from GP or SPs to update the information.
- The Analyst can research the diseases and statistical values based on finalised EMR. However, the analyst does not have permission in current ongoing treatments.

Table 3. List of experimental databases (Stage1 Simulation)

<table>
<thead>
<tr>
<th>DB Name</th>
<th>Encryption(Y/N)</th>
<th>DAS model(Y/N)</th>
<th>DB Characteristics</th>
<th>Record Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain_Hospital</td>
<td>N</td>
<td>Y and N</td>
<td>Plain-text</td>
<td>10K</td>
</tr>
<tr>
<td>Enc_Hospital</td>
<td>Y</td>
<td>N</td>
<td>Microsoft Built-in Encryption scheme</td>
<td>10K</td>
</tr>
<tr>
<td>Enc_Hospital2</td>
<td>Y</td>
<td>Y</td>
<td>AES-DAS</td>
<td>10K</td>
</tr>
<tr>
<td>Enc_Hospital3</td>
<td>Y</td>
<td>Y</td>
<td>AES-DAS (Semi-Bucket)</td>
<td>10K</td>
</tr>
<tr>
<td>Enc_Hospital5</td>
<td>Y</td>
<td>Y</td>
<td>AES-DAS (Bucket plus Bloom filter)</td>
<td>10K</td>
</tr>
<tr>
<td>Enc_Hospital6</td>
<td>Y</td>
<td>Y</td>
<td>AES-DAS (Semi-Bucket)</td>
<td>2M</td>
</tr>
</tbody>
</table>

Table 3 depicts the lists of experimental databases. In the case of a semi-DAS model, the database owners are in charge of preserving the security of their data sets.

In the table 3, AES-DAS (Semi-Bucket) means the partitioning field can be assumed to get meaningful values. And the partitioning field is not encrypted or bit-manipulated.

3.2.2 Stage 1 Simulation Procedure

- Design a prototype EMR on the SQL Server
- Before generating the random data, insert indispensable base records including information about doctors, researchers and nurses to satisfy the constraints
- Operate the Random Data Generator with a range of 10,000 records to 2 million records
- Check the inserted data and the count of the records
- In the case of the 2 million-record database, run the index tuning advisor to satisfy for the query that actually utilises index
- Analyse SQL files in the database query window in MS built-in encryption schemes and perform the Performance Checker inside the Random Data Generator which operates on the AES-DAS and the Plain-text model
- Evaluate the response times of queries from stage 1 simulation result
Table 4. List of experimental databases (Stage 2 Simulation)

<table>
<thead>
<tr>
<th>Record Size</th>
<th>Plain-Text DB</th>
<th>MS Built-in DB</th>
<th>AES-DAS</th>
</tr>
</thead>
<tbody>
<tr>
<td>5K</td>
<td>Plain_Hospital_5</td>
<td>Enc_Hospital_5</td>
<td>Enc_Hospital_BI_5</td>
</tr>
<tr>
<td>10K</td>
<td>Plain_Hospital_10</td>
<td>Enc_Hospital_10</td>
<td>Enc_Hospital_BI_10</td>
</tr>
<tr>
<td>20K</td>
<td>Plain_Hospital_20</td>
<td>Enc_Hospital_20</td>
<td>Enc_Hospital_BI_20</td>
</tr>
<tr>
<td>40K</td>
<td>Plain_Hospital_40</td>
<td>Enc_Hospital_40</td>
<td>Enc_Hospital_BI_40</td>
</tr>
<tr>
<td>80K</td>
<td>Plain_Hospital_80</td>
<td>Enc_Hospital_80</td>
<td>Enc_Hospital_BI_80</td>
</tr>
<tr>
<td>160K</td>
<td>Plain_Hospital_160</td>
<td>Enc_Hospital_160</td>
<td>Enc_Hospital_BI_160</td>
</tr>
<tr>
<td>320K</td>
<td>Plain_Hospital_320</td>
<td>Enc_Hospital_320</td>
<td>Enc_Hospital_BI_320</td>
</tr>
<tr>
<td>640K</td>
<td>Plain_Hospital_640</td>
<td>Enc_Hospital_640</td>
<td>Enc_Hospital_BI_640</td>
</tr>
<tr>
<td>1280K</td>
<td>Plain_Hospital_1280</td>
<td>Enc_Hospital_1280</td>
<td>Enc_Hospital_BI_1280</td>
</tr>
<tr>
<td>2560K</td>
<td>Plain_Hospital_2560</td>
<td>Enc_Hospital_2560</td>
<td>Enc_Hospital_BI_2560</td>
</tr>
</tbody>
</table>

Table 4 depicts the 10-scale-sized databases in the categories of Plain-Text DB, MS Built-in DB and AES-DAS model. The scalability test is to enhance the result of the stage 1 simulation and to make sure the result is also consistent and co-related as the record sizes are bigger. The record sizes are varied from 5K to 2560K. The creation of databases up to 2560K seems to work in the tested machine (Core 2 duo/ Windows XP). If it is over than 2560K, the machine sometimes went crashed. Creating that amount of records is a limit for the machine’s stability. Only one time the simulations are measured because they are plenty of scaled record sized databases.

3.2.3 Stage 2: Simulation Procedure

- Build databases in accordance with 10 scale size databases – 5000, 10K, 20K, 40K, 80K, 160, 320K, 640K, 1280K, 2560K record-sized
  - 5K, 10, 20, 40, 80, 160, 320, 640, 1280, 2560 make double sized compared to previous databases.
- Re-Setup the performance estimation environment among Plain-text, MS built-in, AES-DAS model
- Measure Bulk-insert performance for the three models – Plain-text, MS built-in, AES-DAS model
- Re-check the schema and tune the tables using index tuning advisor
• Measure Read Operation (select query) performance in the categories such as character type, number type, range query, Bucket index range query, aggregation query

3.2.4 Measures / Assessment tools

Query Response times are the primary measurement index. In the case of the Plain-text database and MS built-in encryption scheme, MS SQL Management Studio will be used to measure and assess the response time. For the simulation, the database connecting time is disregarded even though it consumes a part of response time. It is not a major topic of this research. The query optimisers were also carried out to test the queries.

However, it is usually for index tuning not encryption/decryption. The query performed only one on each database.

Besides, two performance checkers were not performed because it would affect the other. Here, in AES-DAS model is an outsourcing model, thus, it limits using the stored procedure. It is recommended in using the core encryption/decryption logic in the application side like a performance checker. It is also easy to check the birth date so as to calculate the specific ranged data in the Bucket index performance.

In the meantime, in the case of AES-DAS or AES-DAS including Bucket index, a custom Performance Checker will measure the query response time. In order to audit the SQLs into DBMS, the SQL profiler is used and checked. The Index tuning advisor tunes the index of each table related to the measurement queries. Additionally, MS Visual Studio 2010 enables the drawing of a class diagram and a sequence diagram for simulation. Therefore, the measures / assessment tools are presented as follows.

• SQL Management Studio: T-SQL measures the query response time and includes the specific categories of queries
  (Only used in the Plain-text and the MS built-in encryption model)
• Performance Checker : Written in C# and measures the query response time and includes the specific categories of queries
  (Only used in the custom AES-DAS model)

3.2.5 Measurement Issues

Before judging the techniques to implement, embedded queries, LINQ (Language Integrated Query Language) and Stored Procedures were tested. The Stored Procedures performed faster than others. However, in the simulated AES-DAS model, the Stored
Procedures are not appropriate techniques. It is because the Stored Procedures are stored in the DBMS side. Therefore, it is not proper to an outsourcing model. However, LINQ provides a lot of functionalities and fill the necessity for the outsourcing model – AES-DAS but not flexible compared to Stored Procedures. Summing up the measurement issues is below.

The measure of execution time could depend on the SQL version. After creation of databases, there is a performance tuning advisor to optimise the queries. Usually, it recommends proper indexes for certain query conditions. In this project, range queries took the longer time than other queries.

In the stage 1 simulation, 6 runs of performance checker are executed. To get more realistic outcomes, in the stage 2 simulation, 30 runs of performance checker in 10 scales are executed.

Two performance checkers can run at the same time but it will affect the DBMS’s performance and H/W resources.

Stored procedures are can be expected to perform faster than non stored queries. In this project, in the case of outsourcing models, the stored procedures are not proper methods because encryption/decryption logic is outside DBMS. In case of AES-DAS, the application logic is outside DBMS and an outsourced DBO does not know how to decrypt a certain specific field in databases.

In the bucket index simulation, it is easy to compare the date time values in range queries. The Bucket index will be key information to judge if the search condition is satisfied or not.

In this project, SQL join is not considered. Because it is another matter of performance evaluation and it consumes extra resources.

Table 5. The measurement issues according to the DB operation environment

<table>
<thead>
<tr>
<th>Database Environment (development)</th>
<th>Plain-text (in-house)</th>
<th>MS built-in encryption (in-house)</th>
<th>AES-DAS (outsourcing)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stored Procedures</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Embedded SQLs (SQL Management Studio)</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>LINQ or ADO(Active Data Object).NET</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
</tbody>
</table>
3.3 INSTRUMENTS

Table 6. Simulation environment

<table>
<thead>
<tr>
<th>Software</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows XP</td>
<td>32-bit Intel Core2duo 3GHz</td>
</tr>
<tr>
<td>C# (.NET 3.5/4.0 Framework)</td>
<td>4GB RAM</td>
</tr>
<tr>
<td>Microsoft SQL Server 2008</td>
<td>500GB HDD</td>
</tr>
<tr>
<td>Bloom Filter</td>
<td></td>
</tr>
</tbody>
</table>

Table 6 depicts the hardware and software environment for the simulations. The source code was initially developed by .NET 3.5 framework and Visual Studio 2010. Microsoft Built-in Encryption hierarchy was designed on SQL Server 2008. In order to simulate the experiments, the modules such as Random Data Generator, Performance Checker and Singleton Designed Bucket Index are implemented. On the other hand, Bloom filter is an open source binary package, so it is difficult to modify inside logic, but provides the concept of this simulation particularly in the AES-DAS using Bucket Index and Singleton Design Patterned.

3.4 ANALYSIS

The main collected data are the query response times; minimum mille-seconds from the stage 1 simulation which has 6 types of databases incorporating with Plain-text, AES-DAS using Bucket Index and MS built-in encryption model. In the case of the MS built-in encryption scheme or Plain-text model, SQL Management Studio measures response times through SQL procedure style phrases using Date and Time functions. In the case of the AES-DAS using Bucket Index model, a Performance Checker measures the pure select query response time except the database connection times by using DateTime classes in C#.

In the stage 2 simulation, the simulation aims for the performance review in accordance with 10-scale-sized databases from 5K records to 2560K records. The scales are increased as double as the previous ones. For this reason, the trend lines for the simulation results among plain-text, MS built-in and AES-DAS can be reviewed.
3.5 APPLICATIONS FOR THE SIMULATIONS

3.5.1 Class Diagram

Figure 18. Class Diagram for Random data generator, Performance checker and Application

Figure 18 depicts the class diagram for the Random Data Generator, the Performance Checker and the AES modules (Rijndal) and the Bloom filter (Singleton).

- Random Data Generator: this application is for bulk inserting in accordance with record size (initial size = 10,000 at Stage 1 simulation, 5K to 2560K at Stage 2 simulation). The Random Data Generator used RijndaelCrypt module when used for the AES-DAS model. In the case of Plain-text, no encryption module was used.

- RijndaelCrypt: this class uses the same symmetric key during encryption and decryption. The framework is called AES. Rijndael’s symmetric key length can be 16, 24, 32 bytes. For this simulation, a 16 byte array of key and iv vector are used.

- Singleton: this class includes a Bloom filter class which is loaded once at initialisation of the object only one time. The Bloom filter reserves the partitioning field information which is a pure non-encrypted field such as birth year in birth date. In the functionality, it converts its values from binary
to integer values so as to give an answer from a request such as “birth year – 1991 is in the Bloom filter?”

- Performance Checker: this class measures the response time in the category of character type, numeric type, range/ rang queries using Bucket Index and aggregate queries in AES-DAS model whereas in other models such as MS built-in encryption scheme and Plain-text use SQL management studio.

### 3.5.2 Sequence Diagram

![Sequence Diagram of AES-DAS using Bucket Index and Bloom filter](image)

Figure 19. Sequence Diagram of AES-DAS using Bucket Index and Bloom filter

Figure 19 depicts the Sequence diagram of AES-DAS using Bucket Index and Bloom filter. This feature is for range query performance evaluation especially. In this project, the range query, the range query using Bucket Index and the comparison between the custom AES-DAS using Bucket Index and the MS built-in encryption.

Firstly, the user sends a request to the Bucket Index Engine which includes a Singleton design patterned collection object – a Bloom filter. In the simulation, the birth date is bucket-indexed, in other words, its key information is partitioned in the database in a secured manner. And the Bloom filter also includes the Boolean values related to the bucket index of the birth date. The Bloom filter only answers “Yes” (true) or “No” (false) when the items in the Bloom filter are found. Thus, if the key information of the query conditions is matched in the Bloom filter, the first phase query is successful and the second phase query is
processed. The second phase query is to grab the encrypted data of the wanted partition only. The main advantage is that the Bucket Index can save search time particularly in range queries because it does not decrypt the whole data.

The implementation methods can vary. The Bucket index can use stored procedures, LINQ (Language Integrated Query Language) or ADO (Active Data Object).NET. However, the Bucket Index is an outsourcing model, thus, the encryption/ decryption logic should be outside DBMS. In this context, LINQ or ADO.NET is more suitable than the stored procedures in the DBMS.

3.5.3 Pseudo Code for Range queries using Bucket Indexes

Table 7. Pseudo Code for Range queries using Bucket Indexes

<table>
<thead>
<tr>
<th>AES-DAS</th>
<th>Pseudo Code</th>
<th>Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td># of Query: 1</td>
<td>10K * DEC</td>
</tr>
<tr>
<td></td>
<td>How to:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>n* DEC</td>
<td></td>
</tr>
<tr>
<td>Semi-Bucket</td>
<td># of Query: 1or 2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>IF Query in BI</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT (n/BI) * DEC</td>
<td>BI: Bucket Index</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(10K/70) * DEC</td>
</tr>
<tr>
<td></td>
<td>IF Query inside DECP</td>
<td>BI is not strong enough</td>
</tr>
<tr>
<td></td>
<td>SELECT (n/BI) * DECP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ELSE IF Query outside</td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELECT (n*BI)<em>AP</em>DECP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RETURN NOMATCHED</td>
<td></td>
</tr>
<tr>
<td>Bucket</td>
<td># of Query : 1 or 2</td>
<td>DECP: bit operation or Vigenere cipher</td>
</tr>
<tr>
<td></td>
<td>IF Query inside DECP</td>
<td>(10K/70) * DECP</td>
</tr>
<tr>
<td></td>
<td>SELECT (n/BI) *DECP</td>
<td>AP: Search Additional Partition(BI)</td>
</tr>
<tr>
<td></td>
<td>ELSE IF Query outside</td>
<td>DECP is stronger than DEC</td>
</tr>
<tr>
<td></td>
<td>SELECT (n*BI)<em>AP</em>DECP</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ELSE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>RETURN NO MATCHED</td>
<td></td>
</tr>
<tr>
<td>Bucket Including</td>
<td># of Query : 0 or 1</td>
<td>BF: Bloom Filter as Singleton designed</td>
</tr>
<tr>
<td>Bloom Filter</td>
<td>IF BF Contains Query</td>
<td>BF in the application side checks Before Talking with DB.</td>
</tr>
<tr>
<td></td>
<td>SELECT (n/BI)*DECP</td>
<td>If data does not exist,</td>
</tr>
<tr>
<td></td>
<td>ELSE RETURN</td>
<td>no additional database connection is required</td>
</tr>
</tbody>
</table>
Suppose that the records are 10K (n = 10K) and the Buckets (the number of the partitions) are 70 (BI = 70), such that DEC means to decrypt by custom AES. DECP means to decrypt module with Vigenere cipher or bit operation, which is stronger than DEC. On the other hand, AP means Search Additional Partitioning and BF means Bloom Filter. Table 7 below depicts the pseudo code for query analysis. Vigenere cipher is for the manipulating the bits of Bucket Index (= Partitioned Key Information). It can be a reversed bits or bits of which the positions are changed. These will make untrustworthy DBO feel difficult in guessing the key information.

### 3.5.4 Bloom filter and LINQ code example

```csharp
try
{
    int k = 0;
    if (Filter.Contains("2000"))
    {
        //string birthDate = "2000-01-01";
        IDerivable <Patient_Identity> identity = context.GetTable<Patient_Identity>().Where(p => p.BirthYear.CompareTo("2000") > 0);
        //string bucketBirthYear = "2000";
        foreach (var v in identity)
        {
            //if (v.BucketBirthYear.CompareTo(bucketBirthYear) >= 0)
            {
                if (RijndaelCrypt.DecryptPatient((string)v.BirthDate).CompareTo("2000-07-01") >= 0)
                {
                    //
                }
            }
        }
    }
}
```

Figure 20. The Bloom Filter and the LINQ Usage

The figure 20 depicts that how the range query is processed in the AES-DAS using Bucket Index which incorporates with the Singleton design patterned Bloom filter.

Firstly, if the Bloom filter contains the birth year such as ‘2000’, then, the Bucket Index Engine roughly gets the data with the same year or the greater than ‘2000’ using LINQ. This is a first coarse query.

Secondly, within the iterate sentence, the Bucket Index can search for more specific dates such as ‘2000-07-01’ using the AES decryption.

Finally, the count of the searched items or the result can be returned to the application.
3.5.5 Pseudo Code for MS built-in encryption scheme

- How to create MS built-in encryption scheme and create a record

```
Create Master Key

Create Asymmetric Key – default RSA 1024

Create Symmetric Key – default TripleDES

Open the symmetric key name

Invoke Encryption Functions – using the key name

Close the symmetric key name
```
• How to read MS built-in encryption scheme and read a record

![Diagram showing the process of reading a record](image)

3.5.6 Sample Codes for MS built-in encryption scheme

- Read Operation using MS built-in encryption scheme

```sql
SELECT uuid, firstName, lastName, cast(DecryptByKey(MedicareNo) as varchar) as medicareNo, phoneNo, cast(DecryptByKey(Address) as varchar) as address, gp_id, pwd, email, regidate from patient_base where cast(DecryptByKey(medicareNo) as varchar) = '1234567890'
```

This query decrypts the encrypted MedicareNo and Address using Microsoft built-in Encryption/Decryption technique with the same symmetric key. MedicareNo’s data type is varchar and content type is numeric. The reason why MedicareNo is varchar is that some MedicareNo may start from 0. It has an intention to keep the number of digits consistently.

- Grant symmetric and asymmetric keys to an authorised user

```sql
GRANT CONTROL ON SYMMETRIC KEY::sym_encryption_my TO ABC (username);
GRANT CONTROL ON ASYMMETRIC KEY::asym_encryption_my TO ABC (username);
```

This grant option is useful when a database administrator wants to give permission to someone who is authorised. However, MS built-in encryption scheme is weighed more on database roles, and in terms of the DAS model, this grant option is not a proper activity.

- Open/Close symmetric key before calling the SQLs
OPEN SYMMETRIC KEY sym_encryption_test
DECRIPTION BY ASYMMETRIC KEY
asym_encryption_test
CLOSE SYMMETRIC KEY sym_encryption_test

⇒ The keys should be opened before use and be closed after use. If the stored procedures or SQL sentences without managing the keys are executed, the SQL Server query processor pops up an error.

3.6 SUMMARY

In this chapter, 3 database operational environments; Plain-text, MS built-in encryption scheme, custom AES-DAS using Bucket Index were introduced. There are 2 simulation stages. The first one is to evaluate various queries including range queries. The second one is to test the scalability which is from 5K to 2560K record-sized on each environment. In this research design, a proto-type EMR is introduced and the encrypted fields such as address, medicare card no, birth date and blood type. The simulations are bound to a PC based and Microsoft development environment. This chapter introduces the class diagram, the sequence diagram, the pseudo codes for the custom AES-DAS using Bucket Index and the Bloom filter and LINQ. Also, it introduces the flow of how to create MS built-in encryption scheme and utilizations through sample codes.

This chapter takes an overall picture of the implementations, proposing as a custom AES-DAS using Bucket Index.
Chapter 4: Simulations and Results

4.1 STAGE 1 SIMULATION

4.1.1. Sample data under the proposed AES-DAS using Bucket Index

The sample data from conducting the Random Data Generator is presented below. This Random Data Generator produces the proposed AES-DAS model which focuses on the Bucket Index and Singleton design patterned with the Bloom filter.

<table>
<thead>
<tr>
<th>Data</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>patient</td>
<td>PID: 1234567890, AES: 1234567890, Bucket: 1234, Character: patient</td>
</tr>
<tr>
<td>shahid</td>
<td>PID: 1234567890, AES: 1234567890, Bucket: 1234, Character: shahid</td>
</tr>
<tr>
<td>doctor</td>
<td>PID: 1234567890, AES: 1234567890, Bucket: 1234, Character: doctor</td>
</tr>
</tbody>
</table>

Figure 21. Sample data of Patient_Base

In figure 21, the uuid means universally unique identifier to avoid the duplicated rows and to be a primary key. This field is the primary key for the Patient_Base table. The MedicareNo is AES encrypted numeric typed data. The Address is AES encrypted character typed data. Pwd is hash functioned processed which cannot be recovered as original values.

This AES-DAS model used the same encryption key for AES. As a result of Select Queries which needs to decrypt the numeric (MedicareNo) and character type (Address), the performance of the numeric type is better than that of character type in the overall results. The detailed result will be presented in the later chapter.

There are three types of Bucket Indexes implemented in this simulation. Figure 22 illustrates the ‘BucketBirthYear’ which is a proto-type Bucket Index. It is a part of birthdate and can be assumed that it represents a year. The ‘BinBucketBirthYear’ which is a semi-Bucket Index as binary values. The ‘BirthBucketYear2’ which is a Bucket Index using Vigenere cipher which is an opposite arrays of binary values. Additionally, the BirthDate and BloodType are AES-encrypted.

10K-record-sized databases were built initially for the simulation that compares Plain-text, MS built-in Security and the AES-DAS using Bucket Index model.

Figure 22. Sample data of Patient_Identity
A further simulation between 10K-record-sized database and 2M-record-sized database for a stress test was compared.

As a matter of the encryption strength, the ‘BirthBucketYear2’ using Vigenere cipher is the strongest. The ‘BintBucketBirthYear’ is the second strongest. And the ‘BucketBirthYear’ is prone to be assumed to real birthdates data.

### 4.1.2 SQL Analyser Tools and Query Plan Results

![Figure 23. SQL Profiler when LINQ is used](image)

Figure 23 depicts how the SQL profiler is monitoring all SQL running on DBMS and can also monitor a specific query at that time and group by different users. The SQL profiler can monitor time-consuming queries or heavily-used queries in DBMS. It also shows the disk I/O for a performance factor. In this simulation, when LINQ is used, it is translated to pure SQLs for DBMS to understand.

![Figure 24. Query Plans for Aggregation Query](image)

Figure 24. Query Plans for Aggregation Query
Figure 24 shows the Query Plans are for analysing Aggregate Query, particularly Blood type. The upper part of the plan is for MS built-in encryption scheme and the lower part is for plain-text. The lowest part of the plan is for AES-DAS.

Table 8. Query Plan Comparison for Aggregation Queries

<table>
<thead>
<tr>
<th>Encryption Type</th>
<th>Query Plan</th>
<th>Coefficients</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain-Text(6ms)</td>
<td>X=0.12*6 = 0.72ms</td>
<td>X=Aggregate(Stream)</td>
</tr>
<tr>
<td></td>
<td>Y=0.88*6</td>
<td>Y=Index scan(Non-clustered)</td>
</tr>
<tr>
<td>AES-DAS(31ms)</td>
<td>X=0.4*31 = 12.4ms</td>
<td>X=Aggregate (HashMatch)</td>
</tr>
<tr>
<td></td>
<td>Y=0.6*31</td>
<td>Y=Index scan(Clustered)</td>
</tr>
<tr>
<td>MS Built-in(80ms)</td>
<td>X=0.8*80 =64.0ms</td>
<td>X=Aggregate (HashMatch)</td>
</tr>
<tr>
<td></td>
<td>Y=0.2*80</td>
<td>Y=Index scan (Non-clustered)</td>
</tr>
</tbody>
</table>

Table 8 depicts the Query Plan Comparison for Aggregation queries. In MS Built-in security and AES-DAS models, the aggregation query consumes Hash Match whereas the Plain-text database uses stream aggregate. As a result, Plain-text database shows the fastest performance than encrypted databases in group by clauses. The Hash Match uses a virtual table in memory to compute extra works such as decryption. In detail, Hash Match constructs a hash table by computing a hash value for rows. After the hash table is built, the query processor scans the table and gives a result. As a matter of index scan, AES-DAS using Bucket Index uses a clustered index whereas the MS built-in security uses a non-clustered index. Usually the clustered index has faster search structure than the non-clustered index. Therefore, the AES-DAS has more efficient way in Aggregation Query compared to MS Built-in security. However, the table for the cost plan shows the X factor (=Aggregate) is more important than the Y factor (=Index scan).
Table 9. Response Time of Insertion Test (10K records)

<table>
<thead>
<tr>
<th>Table name</th>
<th>Patient_Base</th>
<th>Patient_Identity</th>
<th>Patient_Record</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain_Hospital (plain normal data)</td>
<td>18.214</td>
<td>05.773</td>
<td>04.637</td>
</tr>
<tr>
<td>Enc_Hospital (MS built-in Encryption scheme)</td>
<td>195.258</td>
<td>232.161</td>
<td>232.061</td>
</tr>
<tr>
<td>Enc_Hospital2.3.3 (conventional AES-DAS model)</td>
<td>20.780</td>
<td>05.344</td>
<td>05.234</td>
</tr>
</tbody>
</table>

4.1.3 Bulk Insert Performance Result

According to Table 9, MS built-in encryption scheme is much slower than others. Comparing with the custom AES-DAS models, it is 9.4 times slower. Therefore, if records are inserted into the database in bulk, the MS built-in encryption scheme is not recommended. The Patient_Base table took longer than other tables because the Patient_Base is the primary table and sets the Random Data Generation. In the Patient_Base, the Medicare card number and the address fields are encrypted while in the Patient_Identity, the birth date and the blood type fields are encrypted. The Patient_Record field are encrypted in the Patient_Record.
4.1.4 SQL Performance Result

Table 10. Response Times of Select Query Test (milli-seconds)

<table>
<thead>
<tr>
<th>Query</th>
<th>String 1 (Address) (character)</th>
<th>String (MedicareNo) (numeric)</th>
<th>Range Query (BirthDate) (&gt;=)/ (count)</th>
<th>Range Query (BirthDate) Semi/ Bucket Index</th>
<th>Aggregate Query (Blood Type)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain_Hospital</td>
<td>23</td>
<td>23</td>
<td>80/0</td>
<td>Not available</td>
<td>6</td>
</tr>
<tr>
<td>Enc_Hospital (MS built-in)</td>
<td>113</td>
<td>26</td>
<td>200/40</td>
<td>Not available</td>
<td>80</td>
</tr>
<tr>
<td>Enc_Hospital2 (AES-DAS)</td>
<td>156</td>
<td>156</td>
<td>140/140</td>
<td>Not available</td>
<td>31</td>
</tr>
<tr>
<td>Enc_Hospital3 (AES-DAS/semi-Bucket)</td>
<td>156</td>
<td>156</td>
<td>156/156</td>
<td>Not available</td>
<td>31</td>
</tr>
<tr>
<td>Enc_Hospital5 (AES-DAS/Bucket)</td>
<td>156</td>
<td>156</td>
<td>156/156</td>
<td>16</td>
<td>Not available</td>
</tr>
<tr>
<td>Enc_Hospital6 (AES-DAS/semi-Bucket)</td>
<td>249550</td>
<td>2250</td>
<td>31049/31049</td>
<td>3703</td>
<td>Not tested</td>
</tr>
</tbody>
</table>

Table 10 depicts the key result for the Stage 1 simulation. This result is very important because it deals with all models such as Plain-text (unencrypted), MS built-in security (in-house model), custom AES-DASs using Bucket indexes and the Bloom filter. The proposed AES-DAS model has a red-coloured result. The significant result is if the Bucket index is used in a range query, it can be 3.32 times faster than a normal range query operation while the status is encrypted under 10K records and 70 partitioned and Bucket-Indexed.

The Plain_Hospital shows Plain-text that does not use an encryption technique and so is expected to have the fastest performance. The Enc_Hospital is based on the MS built-in encryption technique which is useful for an in-house model. It shows better performance in a simple precise condition such as getting an exact address and a medicare number than the AES-DAS model which is an outsourcing model. However, it shows slower performance in a range or an aggregate functionality than the AES-DAS model.

The Enc_Hospital3 is a semi-bucket index implemented whereas the Enc_Hospital5 is a bucket index model. Therefore, the semi-Bucket Index receives 16 ms in a range query whereas the Bucket Index receives 47ms. (3.32 times faster)

This increase is due to the conduct additional process to encode and decode the birth date using the Vigenere Cipher. Greater analysis on this result is examined in the next chapters.
This simulation result was introduced in the journal [2] David Shin, Tony Sahama, Steve (Jung Tae) Kim, and Ji-Hong Kim (2011). “Data Encryptions Techniques for Electronic Health Record Exchange”; Inn at Laurel Point, Victoria, BC, Canada, February 24-27, 2011. ISSN:1913 2638. in Appendix B: The Abstracts of the publications

4.2 STAGE 2 SIMULATION

The purpose of the Stage 2 simulation is to conduct a stress test or a large scale simulation to gain more practical outcomes. The previous simulation (Stage 1) is based on a 10K record-sized database while the Stage 2 simulation is from a 5K record-sized database to a 2560K record-sized database. This simulation includes 10 scaled databases in accordance with the Plain-text, the MS built-in and the AES-DAS model.

4.2.1 Bulk Insert Scalability Result

Table 11. Bulk Insert result

<table>
<thead>
<tr>
<th>Database name</th>
<th>Patient Base</th>
<th>Patient Identity</th>
<th>Patient Record</th>
<th>Total(s)</th>
<th>Total(Hrs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain_Hospital_5</td>
<td>11.06</td>
<td>6.52</td>
<td>6.03</td>
<td>23.61</td>
<td>0.01</td>
</tr>
<tr>
<td>Enc_Hospital_5</td>
<td>103.8</td>
<td>101.6</td>
<td>103.35</td>
<td>308.75</td>
<td>0.09</td>
</tr>
<tr>
<td>Enc_Hospital_BI_5</td>
<td>10.99</td>
<td>5.97</td>
<td>5</td>
<td>21.96</td>
<td>0.01</td>
</tr>
<tr>
<td>Plain_Hospital_10</td>
<td>18.21</td>
<td>5.78</td>
<td>4.64</td>
<td>28.63</td>
<td>0.01</td>
</tr>
<tr>
<td>Enc_Hospital_10</td>
<td>195.26</td>
<td>232.16</td>
<td>232.06</td>
<td>659.48</td>
<td>0.18</td>
</tr>
<tr>
<td>Enc_Hospital_BI_10</td>
<td>20.78</td>
<td>5.34</td>
<td>5.23</td>
<td>31.35</td>
<td>0.01</td>
</tr>
<tr>
<td>Plain_Hospital_20</td>
<td>42.22</td>
<td>20.97</td>
<td>20.99</td>
<td>84.18</td>
<td>0.02</td>
</tr>
<tr>
<td>Enc_Hospital_20</td>
<td>410.37</td>
<td>407.64</td>
<td>411.62</td>
<td>1229.63</td>
<td>0.34</td>
</tr>
<tr>
<td>Enc_Hospital_BI_20</td>
<td>100.51</td>
<td>23.86</td>
<td>23.74</td>
<td>148.11</td>
<td>0.04</td>
</tr>
<tr>
<td>Plain_Hospital_40</td>
<td>164.62</td>
<td>53.16</td>
<td>58.6</td>
<td>276.38</td>
<td>0.08</td>
</tr>
<tr>
<td>Enc_Hospital_40</td>
<td>908.45</td>
<td>830.56</td>
<td>861.11</td>
<td>2600.12</td>
<td>0.72</td>
</tr>
<tr>
<td>Enc_Hospital_BI_40</td>
<td>202</td>
<td>49.36</td>
<td>50.63</td>
<td>301.99</td>
<td>0.08</td>
</tr>
<tr>
<td>Plain_Hospital_80</td>
<td>605.79</td>
<td>95.28</td>
<td>134.2</td>
<td>835.27</td>
<td>0.23</td>
</tr>
<tr>
<td>Enc_Hospital_80</td>
<td>1857.88</td>
<td>1652.28</td>
<td>1674.3</td>
<td>5184.46</td>
<td>1.44</td>
</tr>
<tr>
<td>Enc_Hospital_BI_80</td>
<td>514.05</td>
<td>98.54</td>
<td>123.32</td>
<td>735.91</td>
<td>0.20</td>
</tr>
<tr>
<td>Plain_Hospital_160</td>
<td>1163.13</td>
<td>227.64</td>
<td>206.56</td>
<td>1597.33</td>
<td>0.44</td>
</tr>
<tr>
<td>Enc_Hospital_160</td>
<td>3724.17</td>
<td>3204.22</td>
<td>3274.95</td>
<td>10203.34</td>
<td>2.83</td>
</tr>
<tr>
<td>Enc_Hospital_BI_160</td>
<td>1313.33</td>
<td>205.19</td>
<td>201.84</td>
<td>1720.36</td>
<td>0.48</td>
</tr>
</tbody>
</table>
Improving Trust and Securing Data Accessibility for e-health decision making by using Data Encryption Techniques

Table 11 presents the results from the Bulk Insertion. The database name with _number means the size of the database. For example, the Plain_Hospital_5 means a 5K record-sized Plain-text database. The Enc_Hospital means MS Built-in encrypted databases. The Enc_Hospital_BI_number means Bucket Index that incorporates the AES-DAS model.

Total(s) means the exact time measured by seconds. Total (Hrs) means the converted time from seconds to hours to build up the database through Bulk Insert.

As a quick result analysis, in the 5K records result, the MS built-in encryption is 9 times slower than the AES-DAS using Bucket Index. However, in the 2560K records result, the MS built-in encryption is only 3.44 times slower than the AES-DAS using Bucket Index.

4.2.2 Plain-text Scalability Simulation

Table 12. Plain-text Scalability Simulation (milli-seconds)

<table>
<thead>
<tr>
<th>Size</th>
<th>String</th>
<th>Numeric</th>
<th>Range</th>
<th>Range-Count</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5K</td>
<td>73</td>
<td>0</td>
<td>186</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>10K</td>
<td>100</td>
<td>0</td>
<td>233</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>20K</td>
<td>160</td>
<td>0</td>
<td>263</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>40K</td>
<td>283</td>
<td>20</td>
<td>246</td>
<td>33</td>
<td>36</td>
</tr>
<tr>
<td>80K</td>
<td>293</td>
<td>13</td>
<td>313</td>
<td>30</td>
<td>123</td>
</tr>
<tr>
<td>160K</td>
<td>493</td>
<td>30</td>
<td>453</td>
<td>30</td>
<td>153</td>
</tr>
<tr>
<td>320K</td>
<td>1030</td>
<td>16</td>
<td>793</td>
<td>53</td>
<td>276</td>
</tr>
<tr>
<td>640K</td>
<td>1753</td>
<td>36</td>
<td>1633</td>
<td>53</td>
<td>476</td>
</tr>
<tr>
<td>1280K</td>
<td>4236</td>
<td>100</td>
<td>3040</td>
<td>416</td>
<td>820</td>
</tr>
<tr>
<td>2560K</td>
<td>7426</td>
<td>50</td>
<td>5060</td>
<td>156</td>
<td>1983</td>
</tr>
</tbody>
</table>
Table 12 presents the Plain-text simulation from 5K to 2560K record-sized database simulation. There are 5 categories such as string, numeric, range, range-count and aggregate query. One of the noticeable things is that the numeric type select query result shows faster than the string (character) type select query result. Chapter 5 details the graphical trend line of the Plain-text scalability simulation result.

4.2.3 MS built-in Scalability Simulation

<table>
<thead>
<tr>
<th>Size</th>
<th>String</th>
<th>Numeric</th>
<th>Range</th>
<th>Range-Count</th>
<th>Aggregate</th>
</tr>
</thead>
<tbody>
<tr>
<td>5K</td>
<td>106</td>
<td>16</td>
<td>200</td>
<td>23</td>
<td>70</td>
</tr>
<tr>
<td>10K</td>
<td>183</td>
<td>23</td>
<td>240</td>
<td>63</td>
<td>56</td>
</tr>
<tr>
<td>20K</td>
<td>316</td>
<td>46</td>
<td>210</td>
<td>76</td>
<td>60</td>
</tr>
<tr>
<td>40K</td>
<td>570</td>
<td>100</td>
<td>350</td>
<td>160</td>
<td>116</td>
</tr>
<tr>
<td>80K</td>
<td>2543</td>
<td>200</td>
<td>830</td>
<td>313</td>
<td>333</td>
</tr>
<tr>
<td>160K</td>
<td>13453</td>
<td>410</td>
<td>1306</td>
<td>636</td>
<td>576</td>
</tr>
<tr>
<td>320K</td>
<td>40806</td>
<td>796</td>
<td>2970</td>
<td>1280</td>
<td>1220</td>
</tr>
<tr>
<td>640K</td>
<td>136600</td>
<td>1656</td>
<td>4830</td>
<td>2513</td>
<td>2423</td>
</tr>
<tr>
<td>1280K</td>
<td>330713</td>
<td>3193</td>
<td>14656</td>
<td>5046</td>
<td>4686</td>
</tr>
<tr>
<td>2560K</td>
<td>718150</td>
<td>26580</td>
<td>27936</td>
<td>10080</td>
<td>8906</td>
</tr>
</tbody>
</table>

Table 13 depicts the MS built-in Simulation from 5K to 2560K record-sized database scalability simulation. There are 5 categories such as string, numeric, range, range-count and aggregate query. The brief results show that after reaching a certain threshold, the response times go increased abruptly.

However, the Range query means retrieving all records matched. The Range-Count query means retrieving only count of the records. Thus, the Range-Count query is much faster than the Range query. Chapter 5 details the graphical trend line of the MS built-in simulation result.
4.2.4 AES-DAS using Bucket Index Scalability Simulation

Table 14. AES-DAS using Bucket Index Scalability Simulation (milli-seconds)

<table>
<thead>
<tr>
<th>Size</th>
<th>String</th>
<th>Numeric</th>
<th>Range</th>
<th>Range-Bucket</th>
<th>Aggregate</th>
<th>Bloom filter</th>
</tr>
</thead>
<tbody>
<tr>
<td>5K</td>
<td>47</td>
<td>16</td>
<td>63</td>
<td>16</td>
<td>31</td>
<td>172</td>
</tr>
<tr>
<td>10K</td>
<td>266</td>
<td>94</td>
<td>156</td>
<td>16</td>
<td>63</td>
<td>188</td>
</tr>
<tr>
<td>20K</td>
<td>344</td>
<td>188</td>
<td>422</td>
<td>109</td>
<td>250</td>
<td>281</td>
</tr>
<tr>
<td>40K</td>
<td>1109</td>
<td>234</td>
<td>641</td>
<td>203</td>
<td>438</td>
<td>531</td>
</tr>
<tr>
<td>80K</td>
<td>3813</td>
<td>344</td>
<td>1313</td>
<td>297</td>
<td>968</td>
<td>1063</td>
</tr>
<tr>
<td>160K</td>
<td>12110</td>
<td>453</td>
<td>2610</td>
<td>453</td>
<td>1328</td>
<td>1938</td>
</tr>
<tr>
<td>320K</td>
<td>40690</td>
<td>656</td>
<td>5172</td>
<td>828</td>
<td>2235</td>
<td>3453</td>
</tr>
<tr>
<td>640K</td>
<td>111585</td>
<td>938</td>
<td>9266</td>
<td>1531</td>
<td>2672</td>
<td>6657</td>
</tr>
<tr>
<td>1280K</td>
<td>289345</td>
<td>1203</td>
<td>19595</td>
<td>2781</td>
<td>5344</td>
<td>13079</td>
</tr>
<tr>
<td>2560K</td>
<td>660851</td>
<td>1875</td>
<td>36987</td>
<td>5226</td>
<td>8704</td>
<td>25986</td>
</tr>
</tbody>
</table>

Table 14 depicts the Bucket Index (AES-DAS) Simulation from 5K to 2560K record-sized database simulation. There are 5 select query response time categories such as string, numeric, range, range-bucket and aggregate plus the Bloom filter loading time. Furthermore, the Bloom filter loading time is added to the simulation because the Bloom filter is related to the Bucket Index Engine.

As a brief result from this table, the range query response times are different when I use the normal range query and the range query using Bucket Index. The Range-Bucket use less amount of decryption time compared to the normal range query in AES-DAS models. However, if the records are less than 5K, the Range-Bucket (using Bucket Index) is 3.94 times faster than the Range (a normal range query) whereas under 2560K record-sized, the Range-Bucket is 7.07 times faster than the Range. This means that the scalability test is critically required when the databases become growing in a large scale. In the mean time, the Bloom filter can be used as an additional feature of the Bucket Index Engine. It can be a Singleton design patterned in the Bucket Index Engine. Its merit is if the data is not heavily changed, the Bloom filter can help to determine the query processes continued and save database connection time when the existence of the searched data is found in the Bloom filter. Thus, the advantages are to be able to save a connection time if the searched data does not exist in the Bloom filter. The disadvantage is to take quite a bit of loading time for the information into the Bloom filter to provide the service.

Chapter 5 details the graphical trend line of the AES-DAS model simulation result. This stage 2 simulation result was introduced in the journal [1] David Shin, Tony Sahama, Steve (Jung Tae) Kim, and Ji-Hong Kim (2011). “Scalability of Encryption Techniques for Electronic Health Record (EHR) Retrieval”, [KIMICS] in Appendix B: The Abstracts of the publications.
4.3 SUMMARY

In this chapter, the 2 simulations were carried out. The Stage 1 simulation focuses on the each category such as string, numeric, range, range-bucket and aggregate functionality under Plain-text, MS built-in encryption which is an in-house model and AES-DAS using Bucket Index which is an outsourcing model. They are all 10K record sized. The Stage 2 simulation focuses on the scalability from 5K to 2560K records to see the trend line for the encryption and decryption process.

As a result from the Stage 1 simulation, the numeric typed data took less time than the string (character) typed data. There are also three types of Bucket Indexes implemented in this simulation. A proto-type Bucket Index, a semi-Bucket Index and a Bucket Index using Vigenere cipher are implemented and compared each other for birthdates. The prototype Bucket Index is the fastest however, it is prone to be assumed. The Bucket Index using Vigenere cipher is a better way to keep the Bucket Index more secured. The main result is that it can be 3.32 times faster than a normal range query operation while the status is encrypted under 10K records and 70 partitioned and Bucket-Indexed.

As a result from the Stage 2 simulation, under 5K record-sized, the Range-Bucket (using Bucket Index) is 3.94 times faster than the Range (a normal range query) whereas under 2560K record-sized, the Range-Bucket is 7.07 times faster than the Range. This means that the scalability test is critically required when the databases become growing in a large scale. It can be concluded when designing the encrypted databases; the scalability needs to be considered.
Chapter 5: Analysis and Discussion

5.1 STAGE 1 KEY ANALYSIS– SELECT QUERY RESPONSE TIMES

Currently, the DAS model is one of the most popular database maintenance models in any industry. One of primary concern of DAS modelling is the responsibility of database security. In the case of semi-DAS models, the responsibility of database security belongs to database owners. Similarly, in the e-health arena, even if the IT tasks are outsourced to outsourced professional database service providers, database security is a primary concern.

In the simulation, there are three types of database management schemes. The Plain-text model can be a DAS or in-house model and its data is unencrypted at all. MS built-in security can also be a DAS or in-house model. However, to concern more on the security issues, it is more suitable for in-house model. If the professional database service providers have top level database system admin roles which mean a system administrator of the database server, he can create a database master key and also know what kind of symmetric keys and asymmetric keys which are used and can give permission to another person to use those keys as well. Therefore, it is not safe enough for MS built-in security to be used in an outsourced semi-DAS model.

On the other hand, an AES-DAS model can definitely be used in a semi-DAS model. The noticeable difference between MS built-in security and AES-DAS is the location of embedded encryption / decryption logic. In AES-DAS, the encryption / decryption logic resides in an application server not in a DBMS server, therefore, the outsourced DBA have no idea what kind of encryption / decryption logic or key values for the encryption are used. However, there can be an issue about the maintenance responsibility role for the application servers and location of them. On the whole, application servers are located in the same back end network with database servers when considering the efficiency of system design and network security. That means application servers are also outsourced to the same professional database service providers. Despite these findings, one merit of this model is that the logic can reside in the application server as a binary format which is hardly understood to humans.

Basically, in this stage 1 simulation, the AES-DAS using Bucket Index is an outsourcing model and the MS built-in security and the Plain-text is assumed to be an in-house model. Therefore, the performance result between an in-house encryption model and an outsourcing encryption mechanism is compared and it needs to be discussed which models are efficient and proper to the scalability.
Figure 25. Response Time of Select query test under 10K records

Figure 26 depicts key results taken from the simulation that measures Select query response times on each different maintenance model. The simulation is under 10K-record-sized databases: Plain-text, MS built-in and AES-DAS models. As a result of the simulation, most Plain-text queries show the fastest response time in each category of queries.

In the case of retrieving String type and numeric type, MS built-in encryption is faster than the AES-DAS model whereas in the case of range query, the AES-DAS model is faster than MS built-in encryption.

However, there is more to investigate in order to bring about more efficient secured EMR (Electronic Medical Record) from the simulation. As an example, string typed data encryption / decryption show much slower response times than numeric typed data encryption / decryption in the MS built-in security and Plain-text compared to the result in the AES-DAS model. Internally, MS built-in security combines both symmetric keys and asymmetric keys. Nevertheless, Microsoft does not fully open its schemes to the public. Nonetheless, based on this result, while using MS built-in security, numeric-typed data encryption / decryption performance is much more efficient than string-typed data and recommended to be used. In this example, the Medicare card number as numeric typed is used for the plain-text data for the MS built-in encryption and the address is used for the plain-text data as string typed.

In the case of range queries based on category: Normal range, Semi-Bucket and Bucket index are used. The Semi-Bucket shows the fastest response time because it does not encrypt the Bucket Index itself. Its Bucket (Partitioned information) can be stored as an
existence of the Bloom filter collection classes. But in the database, the Semi-Bucket Partitioned information field is weak to be assumed to expose key information because the information as a partitioning field can be assumed and acknowledgeable.

In addition, for the simulation the Bloom filter in application memory using Singleton design pattern is used for the first coarse query. If the Bloom filter contains the wanted data, then the real second query is passed to DBMS. The coarse data works by taking the results from the partitioning data or grouping data based on the characteristics of the queries. Therefore, the Bucket index saves decryption processing time due to the narrower coarse results. There are two types of Bucket index: Semi-Bucket and Bucket index. The concept of Semi-Bucket is that a DBA can make logical assumptions of where the readable data will be partitioned to the partitioning field. In contrast, the Bucket index contains hardly-any readable versions of partitioning information that uses binary operation which means conversion to bits and changes its positions reversely similar to Vigenere cipher.

For the simulation, patients’ birth date are used and compared. However, if Bucket index is more complicated, it also takes additional time to calculate arithmetic or binary operation. As a result, the Bucket index performs around 3.32 times faster than a normal range operation.

In the case of group by functionality, if the query does not utilise bounded index, then, it does not show a better performance compared to MS built-in or AES-DAS model. But if the query utilises index scanning, it shows much better performance compared to encrypted models. On the other hand, it depends on the SQL Server query processor to adopt the indexes or there are different aggregate query schemes in accordance with the content of the aggregate query conditions.

When retrieving encrypted alphabetical datafields in the 2 million record database with the DAS model, the performance with nvarchar(max) type as the field ‘Address’ is significantly lower than when using other techniques whereas it is not as prevalent when retrieving numeric data such as ‘medicareNo’.

One of interesting things in this category is the difference between LINQ and embedded SQLs. In most of AES-DAS simulation, LINQ is widely used because it is much easier to connect to DBMS and close the connection compared with embedded SQLs. LINQ supports almost every feature of Microsoft SQL Server and easier handling collection objects. The simulation limits to LINQ to SQL in the case of AES-DAS models. Despite these findings, if the simulation is carried out by using embedded SQLs or stored procedures to resolve the queries, its performance is slightly better than while using LINQ. But this simulation is supported by a new feature of .NET framework 3.x and is considered by avoiding stored procedures because stored procedures are located in DBMS server and do not fit to the concept of AES-DAS model.
5.2 STAGE 1 KEY– PSEUDO CODE ANALYSIS

The detailed algorithm comparison for a range query is below.

Table 15. Stage1 Simulation – Pseudo code & Analysis

<table>
<thead>
<tr>
<th>AES-DAS model</th>
<th>Process (n=10,000 records, # of partition = 70)</th>
<th>Coefficients</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Range</td>
<td># of Query: 1</td>
<td>n* decryption</td>
<td>10,000 * decryption = 156ms</td>
</tr>
<tr>
<td>Semi-Bucket</td>
<td># of Query: 1 or 2</td>
<td>IF R &lt;= unit of partition</td>
<td>SELECT the Ranged data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ELSE IF R &gt; unit of partition</td>
<td>(n*# of used partition / 70) * decryption</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Consume 1st query and return FINISH</td>
<td>R: Range</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cost per row: 112 us</td>
<td></td>
</tr>
<tr>
<td>Bucket</td>
<td># of Query: 1 or 2</td>
<td>IF R &lt;= DECPartitionFunc (unit of partition)</td>
<td>SELECT the Ranged data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ELSE IF R &gt; unit of partition</td>
<td>SELECT the Ranged data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ELSE</td>
<td>Consume 1st query and return</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cost per row: 112 us ~ 329us</td>
<td></td>
</tr>
<tr>
<td>Bucket Including Bloom filter</td>
<td># of Query: 0 or 1</td>
<td>IF BLOOM FILTER Contains</td>
<td>SELECT DECPartitionFunc (unit of partition)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>BLOOM FILTER: in a Singleton designed application heap</td>
<td>DECPartitionFunc(): bit operation</td>
</tr>
</tbody>
</table>
|               |                  | Cost per row: 112 us ~329us | If no condition matched, Retrun directly, Thus, it saves a
Comparing a Bucket Index, the Semi-Bucket Bucket Index is not secured. The Bloom filter saves a database connection when the search predicate is not found in the Bloom filter.

DECPartitionFunct() is to secure the bucket index field which is a partitioned key factor to make the search range narrower. The bucket index field can be simply encrypted by binary operation or Vigenere cipher. In this simulation, the number of partitions is 70. It starts from 1940 and continues to 2010 birth year.

As a result, a Semi Bucket index shows the fastest performance, Bucket index including Bloom filter and finally a normal range query operation which does the whole part of decryption shows the slowest performance.

The bottom line of the performance for these pseudo codes depends on how much the system can decrease the count of the decryption to search the data. In the case of AES-DAS model, the partitioning (Bucket Index) is a well-established technology. In addition, how to organise the partitioning efficiently based on the tasks and the analysis of the EMR is highly important for the performance issues. The database design related to Bucket Index can be optimised to make a better query performance and to be more secured as well.

### 5.2.1 Query Response Times between 10,000 records and 2 million records

![Figure 26. Response Time of Select query test between 10K and 2M records](image.png)

Figure 26. Response Time of Select query test between 10K and 2M records

Figure 27 depicts response times of Select query result between 10,000 and 2 million records. In this figure, when the database size becomes bigger, the performance result between
string and numeric data are different. The string type such as address shows 97.86 times slower performance compared with the number type when the scale is up to 2 million record-sized. Therefore, the AES-DAS model using custom AES is not competent when the source of target encryption field is a long string type.

Figure 27. Logarithmic scale between record size and ch. type string
Figure 28 depicts a logarithmic scale between record size and character type string in the AES-DAS model. As the slope can be seen in the upper part, the slope is steeper than the below which is a record size. In the stage 2 simulation, this scalability is reviewed again.

5.2.2 Performance (Bloom filter vs. HashSet vs. HashTable)

![Performance Ratio among Bloom filter, HashSet, HashTable](image)

Figure 28. Performance Ratio among Bloom filter, HashSet, HashTable

Figure 29 depicts a performance ratio among Bloom filter, HashSet and HashTable. They are all collection classes which can include object type, however, in this simulation; the collected items are string type that represents birth year such as ‘1940’. In the case of HashSet, it stores only distinctive data, so, no duplicated data can be stored. In case of Bloom filter, if it stores properly, it answers yes or no according to its searched collections inside. In case of HashTable, it has a structure like a ‘key=value’. As a result of comparison among those, HashTable has the fastest memory loading time as a Singleton design patterned. On the other hand, the search times among them are overall the same. Thus, in order to implement the Bucket Index engine, the Bloom filter is not only a source for decision-making but also other classes such as HashSet or HashTable can be.
5.2.3 **Bulk Insert Performance (Plain-text vs. MS built-in vs. AES-DAS)**

Figure 29. Bulk insert test on EMR

Figure 30 depicts Bulk insert test on EMR. The MS built-in encryption is an in-house model but the noticeable fact is that the MS built-in encryption technique is not a proper technique when the DBMSs need to have a bulk insert operation whereas the AES-DAS which is an outsourcing model shows similar performance with plain-text that is not an encryption model. Thus, in a business logic that requires heavily insertion into a DBMS, the MS built-in encryption is not a good solution compared with AES-DAS model. However, most of database transactions are based on select query operations. Therefore, this result may not be an important factor to construct an e-health decision-making system.
5.3 STAGE 2 KEY ANALYSIS – SCALABILITY TEST

5.3.1 Bulk insert scalability test

Figure 30. Bulk insert scalability test

Figure 31 depicts the Bulk insert scalability test. When the record size is increased from 160K to 2560K, the processing times are drawn like an exponential curve. The Y axis stands for processing time (Hours). Comparing with Enc_Hospital(MS built-in encryption scheme), Enc_Hospital_BI (Bucket Index) shows much better performance in bulk insertion operation.
Thus, in a very large scaled database, Enc_Hospital_BI(Bucket Index) is a competent method when it is constructed or it needs a batch process like a bulk insert. Plain_Hospital(plain-text which is not encrypted) took the least amount of time to construct the database.

5.3.2 String select query scalability test

![Graph showing scalability test results for Plain_Hospital, Enc_Hospital, and Enc_Hospital_BI.](image)

Figure 31. String select query scalability test

Figure 32 depicts String type select query scalability test. When the record size is increased from 80K to 2560K, the processing times are drawn like an exponential curve in Enc_Hospital(MS built-in encryption scheme) and Enc_Hospital_BI(Bucket Index) whereas
Plain_Hospital (non-encrypted) shows almost consistent result. The Y axis stands for processing time (milliseconds). Comparing with Enc_Hospital (MS built-in encryption scheme), Enc_Hospital_BI (Bucket Index) shows similar performance in String select queries. Thus, in a very large scaled database, the encryption techniques in which the target is string type are a burden.

5.3.3 Numeric select query scalability test

![Figure 32. Numeric select query scalability test](image-url)
Figure 33 depicts numeric type select query scalability test. When the record size is increased from 1280K to 2560K, the processing times are drawn like a very steep curve in Enc_Hospital (MS built-in encryption scheme) whereas Plain_Hospital (non-encrypted) and Enc_Hospital_BI show almost consistent result. The Y axis stands for processing time (milliseconds). In this scalability, there is a threshold point to decrease the database performance in numeric decryption select query. Thus, in a very large scaled database, the MS built-in encryption technique is not competent compared with AES-DAS model. However, compared with string type, numeric type encryption / decryption performance is slightly better. The numeric type encryption in AES-DAS has much more competitive than that in the MS built-in encryption.
5.3.4 Range select query scalability test

Figure 33. Range select query scalability test

Figure 34 depicts Range select query scalability test. When the record size is increased from 640K to 2560K, the processing times are drawn noticeably in Enc_Hospital(MS built-in encryption scheme) and Enc_Hospital_BI(Bucket Index) whereas Plain_Hospital (non-encrypted) increases in arithmetic scale. The Y axis stands for processing time (milliseconds). In this scalability, the results show that the MS built-in encryption scheme is slightly better than AES-DAS model. Thus, the MS built-in encryption technique can be useful when the data is valuable as statistically compared with AES-DAS model. However, in this simulation of AES-
DAS model, the Bucket Index technique was not used. In the next topics, AES-DAS using Bucket Index will be covered.
5.3.5 **Range Count Select query scalability test**

![Range Count Select query scalability test](image)

Figure 35 depicts Range Count Select query scalability test. Comparing with the range query, the select query for birth date counting shows better performance in Enc_Hospital(MS built-in encryption) and Plain_Hospital(plain-text which is not encrypted). On the other hand, Enc_Hospital_BI shows the same performance compared with its Range select query because the encryption / decryption logic is outside DBMS as AES-DAS model. This means that retrieving the whole data and the count of the data takes the same time to compute.
The Y axis stands for processing time (milliseconds). In this scalability, the results show the MS built-in encryption scheme is better than AES-DAS model. However, in this simulation, AES-DAS using Bucket index is not used. In next simulation, AES-DAS using Bucket Index will be covered.

5.3.6 Proposed Range Select query scalability test

Figure 35. Proposed Range select query scalability test

Figure 36 depicts the Proposed Range Select query scalability test. Comparing with Enc_Hospital(MS built-in encryption & Range) and previous simple AES-DAS model, Enc_Hospital_BI(Bucket index with Singleton design patterned Bloom filter) shows much
improvement. The Y axis stands for processing time (milliseconds). This graph can be considered in an outsourcing model and with or without Bucket Index. It shows when the database size becomes in a large scale, the performance difference between AES-DAS with no Bucket Index and the AES-DAS using Bucket Index is significant.

Thus, with this proposed encryption technique (Bucket Index) in AES-DAS model; it is fairly competitive compared to the simple range query select in AES-DAS model.

However, as the record size is bigger, the response times are longer according to arithmetic scale in the AES-DAS using Bucket Index.
5.3.7 Aggregate query scalability test

Figure 36. Aggregate query scalability test

Figure 37 depicts Aggregate query scalability test. The Y axis stands for processing time (milliseconds). The Plain_Hospital shows modest increase according to the record size. However, the Enc_Hospital_BI (AES-DAS model) shows an irregular increase ratio compared with Enc_Hospital (MS built-in encryption). The reason why Enc_Hospital_BI figure is not a stream-lined curve is that it may be related to the performance in encryption/decryption logic.
outside DBMS. In this category, AES-DAS model and MS built-in encryption do not show difference overall.

5.4 DEVELOPMENT DIRECTIONS

According to Figure 25, the conversion to bits of the Bucket index made the Bucket index unreadable. However, the Bucket index can become even more secure using the Vigenere cipher making it more complicated and consequently response times slower. Additionally, these actions took more response time to resolve the Bucket index to identify the original range queries. However, the AES-DAS model with Bucket Index can utilise the DAS model concept fully because its encryption and decryption logic are layered in the application not in the DBMS side. Thus, it is a good technique when the database owner wants database outsourcing model.

For more research, the AES-DAS model can be utilised in a cloud computing or in this context, E-health DACS to share e-health information more freely. The methodologies used to implement e-health DACS can include data-warehousing, web service among the e-health database provider or cloud computing.
5.5 SUMMARY

In this chapter, the result of Stage 1 simulation and the Stage 2 simulation were referred and their trend lines according to the scalability test.

In a brief summary, the result of Stage 1 simulation can determine the following.

- Numeric typed data encryption has shorter response times than Character typed data encryption. Medicare card no and Address were compared.
- The range query using Bucket Index in AES-DAS shows better performance than the range query without Bucket Index in that.
- The scalability test shows the difference between the response times under the condition which is 10K record sized – 3.32 times and those under the condition which is 2560K record sized – 7 times with 70 partitions. This means that the EMR database design needs consideration for the scalability.
- The aggregate query result shows that AES-DAS is sometimes inconsistent performance compared to Plain-text and MS built-in encryption because this AES-DAS’s encryption/decryption logic is outside DBMS.

The compact crucial summary for the suitable database encryption and maintenance techniques are below.

- AES-DAS model (outsourcing model)
  1. Advantages:
     AES-DAS using Bucket Index is much more efficient than a normal AES-DAS when retrieving range conditioned queries.
  2. Disadvantages:
     AES-DAS shows inconsistent performance when retrieving aggregate queries and proceeding the scalability test.
  3. Significance:
     It is important to design the Bucket Index (partitioning) properly, the design can affect heavily onto the specific query performance based on the business tasks.
  4. Range Query Scalability (10K vs. 2560K under 70 partitions)
     - 10K : 3.32 times faster when using Bucket Index
     - 2560K: 7 times faster when using Bucket Index
• Microsoft built-in security model (in-house model)
  1. Advantages:
     If the search condition is exact and is not a range query, the performance is slightly better than the AES-DAS model
  2. Disadvantages:
     Bulk insertion such as initially building up the database takes much more significant time than other models
  3. Significance:
     If the database administrator is trustworthy, the encryption and the decryption module (stored procedure) is much simpler than AES-DAS using Bucket Index
  4. Range Query Scalability (from fastest to Slowest)
     AES-DAS using Bucket Index > MS built-in security > AES-DAS without Bucket Index

• Plain-text model (in-house model)
  1. Advantages:
     It is unencrypted, thus, performance is the best
  2. Disadvantages:
     It is definitely prone to data leak, thus, only suits for an in-house model and takes attention to access control authorities.
  3. Significance:
     No encryption / decryption logic, thus, it is not a proper model for current DBMS utilisation
  4. Range Query Scalability
     Due to no encryption techniques, its performance is the fastest

• Numeric typed data vs. Character typed data
  1. If the data is numeric, its performance is better than that of the character typed data (Medicare card number vs. Address – Suburb)
     This trend happens through all models (Plain-text, MS built-in encryption, AES-DAS model)
Chapter 6: Conclusions

The literature on e-health and data encryption techniques guides how to construct the first simulation which categorises certain type of queries for performance evaluations. In addition, the first simulation raises an issue that the query performance trend would be same in the large scale of databases. Thus, the second simulation focuses on query performances according to the scalability of the databases. The search condition was the character typed data (Address-sururb), numeric typed data (medicare card number), range query among datetime type, aggregate query among blood types. The implementation was built up in Microsoft technology using MS SQL Server 2008 and C# programming language. Through the literature review, the Bucket index is effective in range queries under encrypted. Even though it needs additional key fields or bloom filter and takes more query trials, it is one of methodology in database encryption/decryption techniques under DAS models. The popular DAS model is semi-DAS which means data is operated by database operator but the owner is responsible for the core encryption/decryption logic.

In the simulations, Plain-text shows the fastest performances because it is not encrypted. Based on the results from simulations, Stage 1 simulation focuses on the performance in the categories of string, numeric, range, range using Bucket index and aggregate queries under 10K records. As a result in Stage 1, the noticeable fact is that range queries by using Bucket Index perform 3.32 times faster than normal range queries in AES-DAS which is an outsourcing model. It is a well-established technique that partitions some of the original source data, saves some key information which cannot be assumed easily, keeps the data encrypted in DBMS and uses a Singleton designed collection class such as Bloom filter. However, the MS built-in encryption shows a bit faster results on string and numeric encryption under 10K records comparing with the AES-DAS model.

As a result in the Stage 2 simulation, the scalability test was performed in accordance with the categories in the Stage 1 simulation. The scalability ranges from 5K to 2560K record-sized. In the simulation, the results are usually incremental arithmetically according to the record-size. However, there are threshold points in each category – string, numeric, range and aggregate queries. As an example, in MS built-in encryption, from 1280K record-sized to 2560K record-sized numeric query, the response time increases suddenly in a large step where as in the case of AES-DAS model, the response time increases arithmetically. In the category of normal range operation, MS built-in encryption is slightly faster than AES-DAS model except Bucket Index.
When compared with normal range query scalability, range query with Bucket Index takes much less response time (7 times faster) while the record increases up to 2560K. Therefore, the AES-DAS model with the Bucket Index is the most proposed model for a range query operation as an outsourcing model.

In summary, based on the Stage 1 simulation, the encryption for string type is not more competitive than that for numeric type even though the record sizes are scaled up to 2560K. Secondly, in a bulk insert operation, the MS built-in encryption scheme is not as good as other techniques. Thirdly, for the AES-DAS model, the encryption and the decryption logic should be layered in the application side not in DBMS. Therefore, the Bucket index engine with Singleton designed Bloom filter can be a solution to the AES-DAS model issues. It showed 3.32 times faster than a normal range query operation in an AES-DAS model under 10K record sized.

More research required to study on the scalability of the encrypted databases and the cloud computing. Those can be an option for e-health decision making systems to share information freely because the scalability and the security issues are handled as a top priority in the cloud computing.

Furthermore, machine to machine (M2M) healthcare delivery concept might be useful to explore.
Bibliography


Appendices

Appendix A: The Abstracts of the paper


Electronic Health Record (EHR) retrieval processes are complex demanding Information Technology (IT) resources exponentially in particular memory usage. Database-as-a-Service (DAS) model approach is proposed to meet the scalability factor of EHR retrieval processes. A simulation study using ranged of EHR records with DAS model was presented. The bucket-indexing model incorporated partitioning fields and bloom filters in a Singleton design pattern were used to implement custom database encryption system. It effectively provided faster responses in the range query compared to different types of queries used such as aggregation queries among the DAS, built-in encryption and the plain-text DBMS. The study also presented with constraints around the approach should consider for other practical applications.

URL: http://ocean.kisti.re.kr/IS_mypopo001P.do?method=multMain&poid=kimics&free=


Data encryption techniques to improve health care decision making processes based on the simulation approach are presented in this paper. Database-as-a-Service model (DAS) was utilised to employ the databases and related records for the scenarios selected. Data that was used for the simulation process ranged from 10,000 to 2 million records. When the bucket indexing model incorporated partitioning fields and bloom filters in a Singleton design pattern, it effectively provided faster responses in the range query. The simulation showed that query response time in the bucket indexing model is approximately 3.32 times faster than the normal range query response time in the encrypted database.


In the medical area, the patients’ data is not only their own back data but also valuable large dataset for finding solutions for the diseases. While electronic medical data is becoming popular and used at real work places like hospitals, insurance companies, and major stakeholders such as physicians, patients, it should be securely dealt in any manner. Thus, the best option for keeping it securely is an issue in the area of database security. Some sensitive data of it should be encrypted in the database. There are many encryption/decryption techniques and algorithms. These days, those are also required to come to an end with performance. The proposed system includes the design of the service scenario, and metadata model that will enhance the data accessibility and facilitate speedy indexing and querying. This document defines the concepts of various medical solutions, the fundamentals of major encryption schemes, the concepts of distributed applications and built-in DBMS security and programmability.

URL: http://ocean.kisti.re.kr/IS_mypopo001P.do?method=multMain&poid=kimics&free=


EMR (Electronic Medical Record) is an emerging technology that is highly-blended between non-IT and IT area. One of methodology to link non-IT and IT area is to construct databases. Nowadays, it supports before and after-treatment for patients and should satisfy all stakeholders such as practitioners, nurses, researchers, administrators and financial department and so on. In accordance with the database maintenance, DAS (Data as Service) model is one solution for outsourcing. However, there are some scalability and strategy issues when we need to plan to use DAS model properly. We constructed three kinds of databases such as plain-text, MS built-in encryption which is in-house model and custom AES (Advanced Encryption Standard) – DAS model scaling from 5K to 2560K records. To perform custom AES-DAS better, we also devised Bucket Index using Bloom Filter. The simulation showed the response times arithmetically increased in the beginning but after a certain threshold, exponentially increased in the end. In conclusion, if the database model is close to in-house model, then the vendor technology is a good way to perform and get response times in a consistent manner. If the model is DAS model, it is easy to outsource the
database, however, some technique like Bucket Index enhances its utilization. To get faster query response times, designing database such as consideration of the field type is also important. This study suggests cloud computing would be a next DAS model to satisfy the scalability and the security issues.