Sudan University of Science and Technology

College of Graduate Studies

College of Computer Science and Information Technology

Title:

Performance Analysis of Classical Encryption Algorithms in encrypting Database Transactions

تحليل أداء خوارزميات التشفير التقليدية في تشفير المعاملات على قواعد البيانات

A thesis submitted in partial fulfillment of the requirement of M.Sc Degree in Computer science

By: Mohammed Suleiman Mohammed Rudwan

Supervised by: Dr. Salah El-Dinn Deng AlJack

March, 2016

الآيـــة

بِسۡمِ ٱللَّهِ ٱلرَّحۡمَٰنِ ٱلرَّحِيمِ

ٱقۡرَأۡ بِٱسۡمِ رَبِّكَ ٱلَّـذِى خَلَقَ ﴿1﴾ خَلَقَ ٱلۡإِنسَٰنَ مِنۡ عَلَقٍ ﴿2﴾
ٱقۡرَأۡ وَرَبُّكَ ٱلۡأَكۡرَمُ ﴿3﴾ ٱلَّذِى عَلَّمَ بِٱلۡقَلَمِ ﴿4﴾ عَلَّمَ ٱلۡإِنسَٰنَ مَا
لَمۡ يَعۡلَمۡ ﴿5﴾

صدق الله العظيم

سورة الفـلق (الآيات 1 – 5)

Dedication

I dedicate this work

To my parents,

To all knowledge seekers and providers,

To all my teachers and instructors,

To all my colleagues,

And to my all friends and classmates.

Acknowledgement

First I'd like to thank God – Allah for what I achieve and completing this research.

I'd like also to express my deepest thanks to my thesis advisor and supervisor Dr. Salah Eldeen Deng Eljack Assistant Professor at the college of computer Science, University of Bahri. He kept advising me and correcting mistakes I made while doing this research

I'd like to grasp this opportunity foremost to express my greatest thanks to all who have helped me towards the successful completion of this research.

Beside those, I cannot express enough thanks to Dr. Awad Mohammed Awad Elkarim, Associate Professor at University of Tabouk for his valuable motivation and advise to start this research.

Last but not least, I have to confirm that my completion of this project could not have been accomplished without the support of my family especially my father, my colleagues at work, and also my classmates.

Mohammed Suleiman Mohammed

Abstract

Database Transactions are pieces of information which holds operations that are performed on the database. They became a very critical issue that should be carefully stored and secured while they are under process, and during their existence in memory.
In this research, encryption was introduced as a solution for securing database transactions and text data that it holds in memory. The main objective is that to measure the performance for three algorithms: Simple Substitution, Caesar, and Periodic Permutation respectively. Performances are measured here in terms of time, memory usage and CPU consumption. Final results we had obtained defined clearly – in terms of time – that in large data processing, Caesar and periodic Permutation had the optimal performance while Caesar had recorded the best score than the other two in case of small data sizes. On the other hand, and in terms of memory usage, simple substitution algorithm had recorded the smallest size used to perform in small-sized data, but in large data, Caesar had won the competition over them. From CPU resource point of view, Simple Substitution had recorded the least efforts of CPU in small-sized data and the same in large-sized data, while Caeser had consumed largest CPU efforts in large as well as small sized data. All in all, this experiment can be based to enable Database Designers to decide which algorithm is the suitable one according to type of database transaction waiting for processing, as well as length of data that it consists of.

المستخلص

تـُعرف العمليات على قواعد البيانات على أنها ذلك الجزء من المعلومات الذي يحتوي على العمليات المختلفة والتي تطبق على قواعد البيانات. لقد باتت من القضايا المهمة والتي يجب أن يتم الاعتناء بطرق تخزينها و استرجاعها وضمان أمنها. لذلك ، العمل على ضمان أمنها أثناء وجودها بذاكرة الوصول العشوائية هو قضية يجب إيلاءها اهتمام ما. لقد تطرقنا في هذا البحث للتشفير كحل من الحلول لضمان تأمين عمليات قواعد البيانات في الذاكرة وكذلك البيانات النصية التي تحتويها. إن الهدف الرئيس من هذا البحث هو قياس أداء خوارزميات : قيصر ، الإحلال البسيط ، وخوارزمية البعثرة وخلط الحروف. تم تناول قياس الأدء من حيث الوقت المستغرق لإنجاز العملية المذكورة لكل خوارزمية على حدة، كما تم قياس الأداء لذات الخوارميات من حيث مساحة استهلاكها للذاكرة ونسبة استخدامها لوحدة المعالجة المركزية . أوضحت النتائج النهائية التي تم التوصل إليها وبشكل واضح أن خوارزميتي قيصر ، والخلط وبعثرة الحروف الدورية كانتا الأفضل من حيث سرعة التنفيذ الزمني من حيث ضخامة حجم البيانات. أما في حالة البيانات صغيرة الحجم ، فقد سجلت خوارزمية قيصر المرتبة الأولى من حيث قصر زمن في تشفير البيانات المستهدفة. أما من حيث استهلاك الذاكرة فقد سجلت خوارزمية الإحلال البسيط أقل مساحة تخزينية مستخدمة للتنفيذ في حالة البيانات صغيرة الحجم، أما في حالة البيانات الضخمة فقد تقدمت خوارزمية قيصر عليهم. أما من حيث استخدام مورد وحدة المعالجة المركزية فقد جاءت خوارزمية الإحلال البسيط مرة أخرى في المرتبة الأولى في حالتي البيانات صغيرة وكبيرة الحجم ، أما خوارزمية قيصر فقد كانت المستهلك الأكبر من بين الخوارزميتين الأخراوتين لمجهود وحدة المعالجة المركزية في حالتي البيانات صغيرة وكبيرة الحجم . أخيراً ، تعتبر تجربة هذا البحث أساس لمصممي قواعد البيانات لتمكينهم من اتخاذ القرار المناسب في شأن الاختيار من بين خوارزميات التشفير المذكورة آنفاً لتشفير عمليات قواعد البيانات حسب نوعها ، وحجمها وحجم البيانات التي تحملها.

Table of Contents

List of Figures

List of Tables

Abbreviations

Acronym	**Meaning**
DBMS	Database Management System
DBA	Database Administrator
AFIM	After Imaging
BFIM	Before Imaging
TDE	Transparent Database Encryption
CPU	Central Processing Unit
RAM	Random Access Memory
HDD	Hard desk
SSL	Secure Socket Layer
SSH	Secure Shell
TDE	Transparent Data Encryption
ACID	Atomicity, Consistency, Integrity, Durability
NTFS	New Technology File System
MH	Mobile Host
FH	Fixed Host
AED	Advanced Encryption Standard
SQL	Structured Query Language

CHAPTER ONE

Introduction

Chapter (1)
Introduction

1.1 Research Background

Using Encryption in databases usually considered as a last line of defense against illegal access to those databases. However, So as to guarantee the confidentiality of the in-memory database transactions, classical encryption algorithms are suggested to be used for doing so. Therefore, comparing those encryption algorithms to encrypt such transaction will be an essential issue that DBMSs designers should maintain carefully in order to choose the right option in terms of time that it takes. Many factors no-doubt affect the processing speed of the encryption algorithm, so the researcher determined a specific specifications of the processing environment and tested different lengths and sizes of subject transactions. The research could open eyes and minds for future researches that can manage to make the decision on making the selection criteria can be variable according to each state of the processor and any other factors that could affect the performance of the processing such transactions through special algorithms can be designed for such purpose.

1.2 Research questions

Having introduced such a subject, the question is that what encryption algorithm should the Security Administrator, the DBA, or the application used to encrypt such database transactions in terms of time?

The question is actually follows into two sub-questions:

1. Which encryption algorithms will be optimal to encrypt certain type of transactions?
2. Which one of them will be the optimal in terms of length of database transaction and data it holds?

1.3 Problem Statement

The problem of this research is to determine the performance of different encryption algorithms in encrypting database transactions and how it affects the CPU time to perform such encryption.

1.4 Research Motivation

Having observed the vast area of movement and the high hit ratio of reaching the servers illegally by hackers, The researcher makes another defense layer that prevents those hackers from understanding what actual database transactions being stored at the log buffer are requested. And as System administrators know, resources should be utilizes so that to perform balance load without affecting too much the time consumed for processing database transactions. Because of that the researcher found this research idea.

1.5 Research Scope

The research will just shows the performance and time taken for processing of each pre-mentioned algorithm. Also, a comparison between them in terms of time will be performed given the environment variables specifications.

Only text data are under study. Hence, vast area of characters are considered as a choice that we may expect to be contained within the 'data' part of different transactions.

1.6 Main Study variables:

a. Type of transaction (read-only, read-write, commit, abort, and start)
b. RAM size, RAM type
c. Processor Speed, processor type
d. Speed of HDD read/write
e. Length of data to be encrypted

1.6 Research Objectives

The objectives of this research are to:

1. Filter the encryption algorithms that are adequate for encrypting read-write and read-only transactions from other types making reasoning for such opinion.
2. Make a comparison in terms of time between the candidate encryption algorithms, namely, simple substitution, Caesar, and periodic permutation.
3. Give researchers and DBMSs designers a principal consideration about the performance of each encryption algorithm so that to fit them to suitable

processing units such as processor, RAM, … etc when encrypting them in-memory for security purpose.

1.7 Hypothesis

All transactions that are about to be encrypted will be assumed that they are syntactically correct. And the algorithms will be perform their operations to each part of the transaction parts separately.

As well, in order to encrypt such transaction, it is assumed that the transaction is correctly and completely arrived to the database server waiting for encrypting before getting executed.

1.8 Research Structure

This study is broadly organized into five chapters, in chapter (1) we gave a brief introduction . Chapter (2) for scanning and discussion of literature review . Chapter (3) contains an explanation of the methodology of performance analysis of algorithms under study . Chapter (4) discusses the results and discussion. And finally, Chapter (5) contains a brief conclusions and recommendations for the future work.

CHAPTER TWO

Background and Literature Review

Chapter (2)
Literature Review

2.1 Encryption & Cryptography

"Cryptography is the science and study of secret writing" [1]. Usually, secret writing refers to making some changes to a message so that it cannot be readable for others except to the authorized ones.

The process of transforming the original message or the text one is attempting to send to another party – called the plaintext or clear text– into the modified form – called the cipher text or cryptogram – is called encryption [1]

Usually, the process of encryption require a private, secret method to perform the transformation of the plaintext into ciphertext, this method is called the cipher [1].

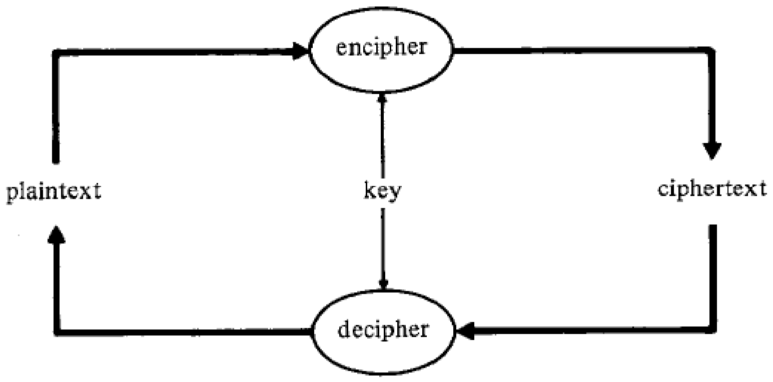

Figure 2.1: Secret writing using encryption and decryption*[1]*

We have two types of encryption, either symmetric in which the encryption key is the same decryption key, or asymmetric in which the encryption & decryption keys are different .

2.2 Substitution and Transposition

Transposition and Substitution are early classical types of encryption. They are usually employed to do the securing process in text-based data.

Transposition method reorders the characters or the bits of the plaintext without having to insert any new bits or character to the cipher text. Whereas Substitution method replaces characters or bits of the plaintext with other ones different from the original ones at the plain text.

2.3 Cryptanalysis :

"It is the science and study of methods of breaking cipher" [1] . It has many ways to perform cryptanalysis. Each way is dedicated to know the keys and the ciphers that are used once the cipher text is sent to other parties. Some ways consider the statistics of letters in a certain language, others can depend on the size of ciphertext being sent so that to guess which attributes of a specific database and table is sent, and so on.

2.4 Database Encryption

Database Encryption usually considered as the last layer of defense in the whole security system that is implemented by the Security or the Database Administrator, but it costs a lot[3]. By this way, even the network and the server was breached by intruders and hackers (cracker) and gained full privileges to do everything with data that is stored in the database, he or she can't make use of that data because they are not in their formal form, but in encrypted form**.**

However, database encryption falls into two main categories: Encryption of data-in-transit, and encryption of data-at-rest. The former one is concerned with securing the related data being transmitted from client to server and vice versa using many techniques for doing so such as Secure Socket Layer(SSL), Secure Shell (SSH), or even using database-specific features that are available in DBMSs…etc. The latter aims to encrypt the data of the database at its end point – at the database environment ; where it exists [3] , it can be classified into two divisions: firstly, Transparent/External Data Encryption(TDE) in which the encryption is done in the entire database and it can be performed through OS/File system. Secondly, User/Data Encryption, in which the encryption is done on specific columns within a table or even on a specific data element or data item. [7]

2.5 Performance Degradation as an effect of encryption at the application layer

Encryption on the application layer usually makes "the encryption/ decryption code may need to be written in multiple locations using multiple libraries, making the solution difficult to implement and maintain" [3] and this takes time no-doubt, However, in order to verify privacy and confidentiality we may sacrifice the performance a little bit. All in all, DBMSs designers should consider the option of encryption as a must, especially in critical systems, as well, they should consider the job-mix while they are injecting the encryption of the data property; that is to consider best practices on applications design and development, algorithm design, and physical properties.

2.6 Options for implementing encryption at data-at-rest

There are three options for encrypting data-at-rest: Encryption at the application layer, encryption at the file system layer, and encryption within the database[2-3].

Options on where to implement the encryption process differs, one option can be encrypting data at the application layer where the encryption process is performed in the application then the data is transferred to the data files. The other option is that to encrypt the data at the file system layer where an advanced file systems for storing data on the disk are configured and used to store such data in an encrypted form. The third option is that to encrypt the data within the database where the encryption methods are embedded in the database server engines and processors. [3]

2.7 Database Transaction

A database transaction is “an executing program that forms a logical unit of database operations that forms the database transaction”[5]. It was found so that to keep the privacy of each transaction and avoid faults that can occurs due to unreasonable erroneous interleaving between actual database queries.

Each transaction should have the property of ACID – Atomicity, Consistency, Integrity, and Durability.

Each database transaction has information about it so that to keep track of each database data entry or updates. Possible Information stored for each transaction includes: type of transaction, transaction ID, data item accessed, Before Imaging value (BFIM), After Imaging value (AFIM), Page ID, length of update item, and the item’s offset from the beginning of the page. The first two pieces of data is mandatory for each transaction. Data item accessed information is stored if the type of transaction was either read or write. The rest mentioned pieces of information are always stated and stored as transaction information in case the type of transaction was read-write transactions, not read-only transaction.

2.8 Log buffer and the system log

As the heritage says, each program attempts to execute should be loaded to memory. And so that the database transactions. Database Transactions are usually stored in the special area of the memory that is reserved by the DBMS in the startup moment. In this area, there is a specific area that is employed for storing each database transaction information before reflecting its actual result into the data file at the disk drives. This area is called log buffer.

Whereas the system log stores all database transactions in repository called system log , or sometimes it called the log. They are stored in a specific form as follows, all information is confined between two brackets, ‘[’ and ‘]’. And each piece of information is separated from others by comma

2.9 “A Survey on Data And Transaction Management in Mobile Databases” study

This section gives an overview and discussion about the paper titled above “A Survey on Data And Transaction Management in Mobile Databases"[6]. Next subsections the researcher will give a quick overview about database mobile management issues as the said paper above discusses and its architecture. As well, we will give a glance on the challenges that the mobile database transactions face as conclusion and discussion about such study.

2.9.1 Overview of Mobile Database

Mobile Database is a database which is stored on any mobile device [7] [3] . A mobile device is any device that can be removed from place to another easily such as Cell phones like Nokia, Samsung, ..etc, Laptops, PDAs, Tablets and Notes. Mobile databases are executed in distributed manner which may be subject to further restrictions due to inherent constraints such as limited bandwidth. Moreover, distribution of transaction in the presence of mobility makes transaction commitment and termination quite complex, they have to ‘forced wait’ or ‘force abort’ because of the lack of wireless channels, and further it may be delayed due to random hand-offs.

2.9.2 Types of Mobile Database Architectures

There are several architectures that could be used for mobile databases based application, they varies in the way of processing the mobile requests; communication for sending and receiving Internet data for example, or querying the database.

They include : client-server architecture, Server/Agent/Client Architecture, and Peer-to-Peer Architecture.

In client-server architecture there are three main components : a server, fixed hosts and mobile clients. When a mobile enters a shadow area of wireless network it will be in a disconnected state then the mobile host which has already hoarded from the server to process transaction locally during disconnections. Once wireless connection is detected and connected successfully once again, then the mobile client connects with the server then incorporates effects of the committed transactions during disconnection into the server database. Figure 2.1 illustrates the client-server Architecture for Mobile databases.

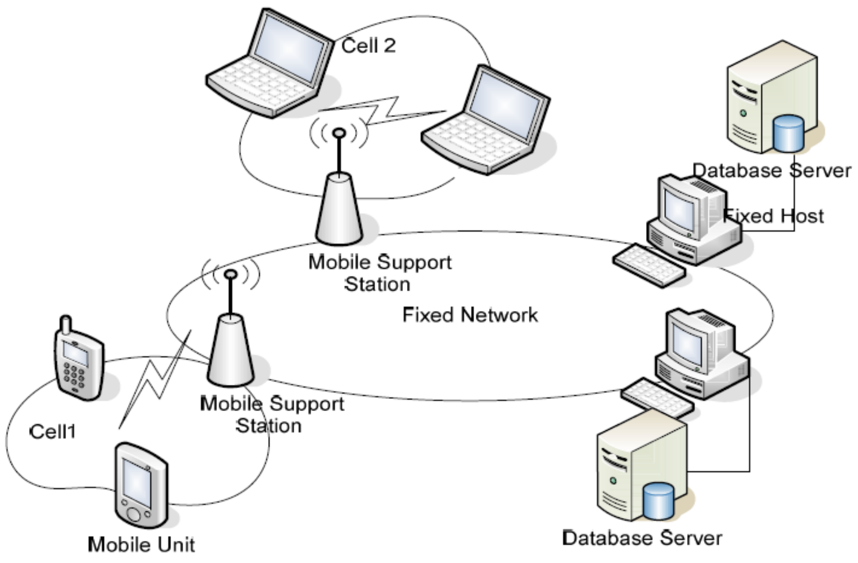

Figure 2.2: Client-Server Architecture for Mobile databases[6]

Other architecture that the paper discussed is that Server/Agent/Client Architecture in which an agent is placed either in the server or in the client partyThe architecture of the mobile database based on agent includes three major layers, namely, Server layer, Mobile Agent layer, and Mobile Terminal layer. Each of which has some agents that perform a specific tasks. Figure 2.1 Illustrates the Server/Agent/Client Architecture.

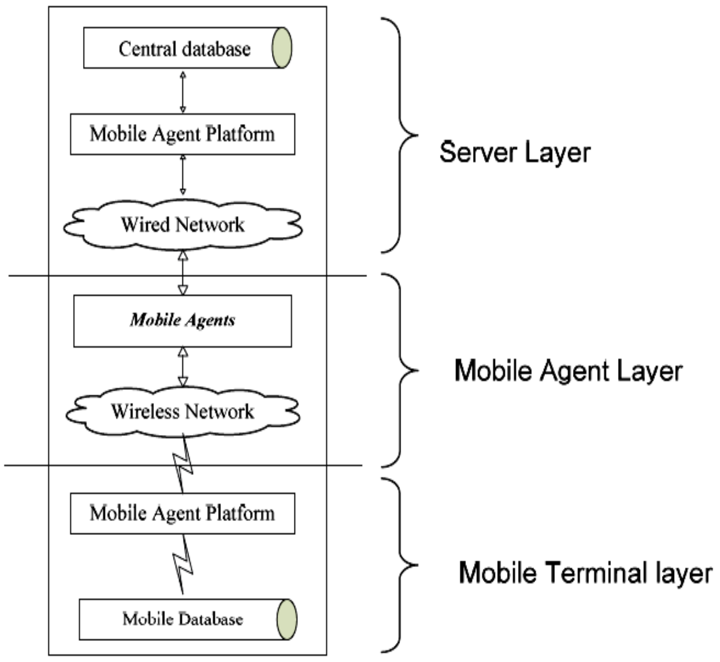

Figure 2.3: Server/Agent/ Client Architecture for Mobile databases*[6]*

A third Architecture that was discussed is peer-to-peer architecture in which clients communicate with other clients in order to share their data with them. A positive point of this is that the high level of availability is met.

2.9.3 Mobile Transaction Models

There are many models found to process mobile transactions according to the paper mentioned in section 2.9 such as initiated as well as Executed Mobile Host(MH), Initiated at MH but Executed by Fixed Host(FH), and Initiated at MH and Executed among MH and FH. Mobile transaction has different characteristics from the usual database transaction, they include:

- They should carefully support and handle concurrency, recovery, disconnections and mutual consistency of the replicated data.
- They are long-lived because of mobility of both data and users as well as multiple disconnections.
- Tasks that each part of each server that mobile connects to should be carefully maintained.

2.9.4 Caching and Query Processing

Caching is an efficient way to improve the throughput of the execution of queries. The idea presented in the same paper mentioned in section 2.9 and which we are discussing these subsections, its idea is that to perform the tasks in the local machine of the client, later on, the update can be actually done and affected in the database center as soon as a connection is available.

Many Protocols can be used to manage caching (replacement policy) such as : Quientet structure, RBF-FAR (check its abbreviation) Query Graphy Model, S-COAS and SAIC.

2.9.5 Recovery in Mobile databases study

The author(s) of this study; the pre-mentioned paper in section 9.5, stated many protocols are available for performing recovery in Mobile databases. For instance, checkpoint protocol with its different versions and variations can be used for such a purpose. We have Message Login Protocols as well. So one of the reviewed papers discussed the mobile database security as such paper[5], and showed that securing mobile database requires securing the three triangle points: the mobile network, the operating system on the mobile device, and the mobile device itself.

2.9.6 Security in Mobile databases

At the section titled “Security in Mobile databases” the author of the paper under discussion; i.e, [6], they reviewed several papers and studies that discussed database security issues. Most of them focused on the access control rather than database contents security.

2.10 “Performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security” study

In this section we will discuss the study title “performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security” [8]

2.10.1 Overview of the study

The mentioned study; i.e.,[11-14], gave an overview of Performance of the Encryption Algorithms, namely, AES128, AES192, AES256, 3DES168, are measured when encrypting databases in cloud using the concept of TDE (Transparent Data Encryption) which is a method for efficient encryption of data at the rest, on hard drives, and data in-transit as well as on backup media. It is well known that Oracle and Microsoft SQL server are the DBMSs which applies the TDE .

The authors didn't forget – on the other hand – to reference to the effects of encryption on the overall system throughput which were classified into two main categories: Query processing performance, and key management headache. And that was the motivation of the authors to write the paper under review here in.

The paper also cited the results of the performance of the said Encryption algorithms above in statistical diagrams representation, it also shows clearly differences between each of them in terms of time, size of memory they took, and CPU efforts consumed to perform the encryption by each.

2.10.2 Types of Transparent Data Encryption (TDEs):

The paper study as we referenced to in section 2.10 above discussed two main types of Transparent Data Encryption: TDE Table-Space encryption, and TDE column encryption. The former works on the entire tablespace and also the entire object created by in the tablespace. The latter encrypts specific columns (attributes) within a relation or table.

SQL supports both types of TDEs, Table2.1 illustrates a sample of possible command that adopted TDE.

Table 2.1: SQL Command quick reference in TDE[8]

Task	SQL Command
Wallet setup in sql.ora	ENCRYPTION_WALLET_LOCATION = (SOURCE= (METHOD = FILE) (METHOD_DATA = (DIRECTORY = C:\APP)))
Initializing database Master key	ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY “AMINU9090”;
To *OPEN* wallet	ALTER SYSTEM SET ENCRYPTION WALLET OPEN IEDNITFIED BY AMINU9090;
To *CLOSE*	ALTER SYSTEM SET ENCRYPTION WALLET

	CLOSED IEDNITFIED BY AMINU9090;
Add encrypted column to existing table	ALTER TABLE table_name ADD (column_name datatype ENCRYPT
Encrypt unencrypted existing table	ALTER TABLE table_name MODIFY (column_name ENCRYPT)
Create Table and encrypt column	CREATE TABLE <table_name> (column_name datatype ENCRYPT)
Creating table from an existing table	CREATE TABLE <table_name> AS SELECT * <existing_table > ;

2.10.3 The experiment

The experiment was performed in three samples' categories of data, firstly, the one column of the whole records, secondly two columns, lastly three columns are considered to be encrypted using various encryption algorithms.

The following diagrams are the final results of (the average) time taken to perform the encryption process, size that each encryption algorithm took, and finally the degree of efforts that each algorithm consumed in order to perform its task . There are three statistical diagrams, the first one shows the results of encrypting algorithms when encrypting only one column, namely, the salary column, the 2nd one shows the results of encrypting 2 columns, the 3rd one is for the results of encrypting three columns of the given table of size 665060 Bytes.

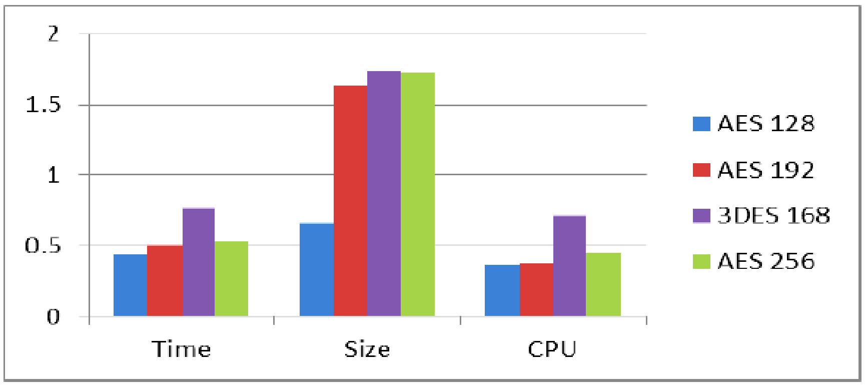

Figure 2.4: Performance result of each encryption algorithm for Salary column*[8]*

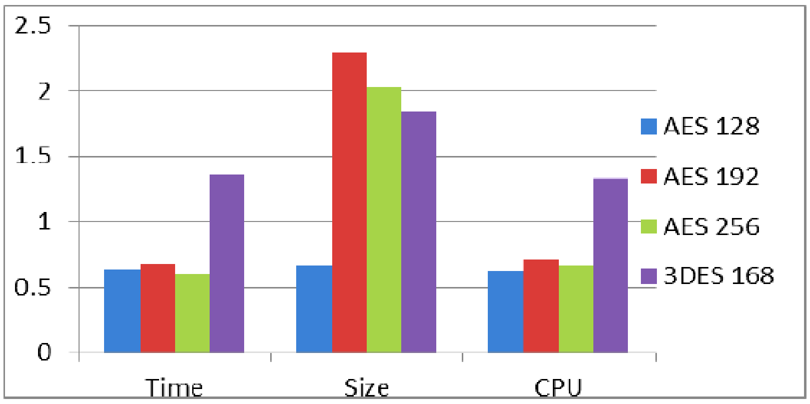

Figure 2.5: Performance Result of each encryption algorithm for Salary and Job Columns*[8]*

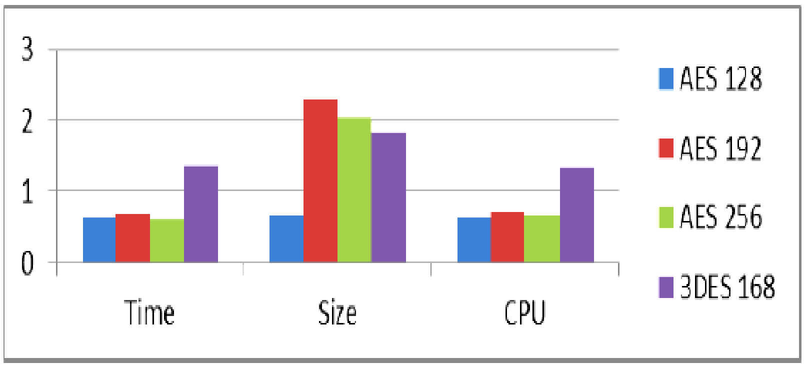

Figure 2.6: Performance result of each encryption algorithm for Salary, Job, and EmpNo columns*[8]*

All in all, results found are that the algorithm AES128 had scored the biggest positive points, it shows that it took the least time to perform the encryption process, as well, the size of memory it took in the three cases (sample groups) are the smallest. Also, it didn't use much CPU effort for execution.

2.11 Discussion & Background

More studies that are reviewed are survey or review papers, some of them discussed the performance of securing database contents by encrypting them while they are existing in the database or encrypting or securing data (transactions or other database data) in-transit. No previous studies are found that discussed in details and in a specific way the issue of securing database transactions in memory. On the other hand, Software

Security science usually discusses securing the memory from illegal access or change. So, Securing memory contents needs more and more studies.

In section 2.1 we gave an overview of the basic concepts that the research is based on its ideas, that s the encryption and cryptography terms.

In section 2.2 we discussed more in-depth the details the classical encryption methods which are transposition and substitution. In section 2.3 we gave and overview of the cryptanalysis in order to show to which degree of importance that it can have while encrypting data. Section2.4 discusses talked about database encryption. In 2.5 we discussed how the encryption can affect the overall performance. Whereas section 2.6 gives the possible options for encrypting data-at-rest . Then sections 2.7 & 2.8 give comprehensive talk about database transaction and log buffer respectively so that to be aware of the nature of what called database transaction as well as its usage in DBMSs.

In section 2.9 we discussed the study – a publish paper [6]– that presents a survey on data and transaction management in mobile databases. As we know, mobile databases have special cases the requires special, advanced methods to handle its services and to overcome errors while moving from one network area to another and the processing is done in a distrubted manner. Mobile DB and transaction faces several difficulties becase of big possibility of errors occurrence, . The researcher came over this study because the scenario will be more complex, and the degree of danger in processing transactions in memory in mobile databases is greater than what we faced in tranditional database servers because the transactions will resident in memory longer time than other database servers as a result of random, multiple hand-offs and wireless connection errors. The study refers to that 45% of attacks are from insiders, so the role of encryption will be more and more important. And according to [5] and other studies as the study showed, no comperhensive solutions are typically found yet as well as no practical solutions are applied and got use of in real industry, so the need of research in this area still needs efforts and contributions of both the scientific researchers and the industry.

In Section 2.10, we discussed the study titled “performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security” [8]. It seems to be same as subject of this research. It discussed some encryption algorithms in terms of performance in encrypting the actual database data, specifically, the attributes (the columns) of a relation using TDE – an efficient encryption method of database contents as well as data in-transit. TDE can protect data in column level, table level, and tablespace level. The study applied the encryption algorithms using TDE in column level. Examples of some leading DBMSs supports such a method in encrypting their data. As the study shows, the main problems that the author(s) found in encryption process are : performance, and keys management. As cloud computing evolved and its usage is dramatically increasing, necessarily securing the end point of data is a must. Evolving of cloud computing can clearly be stated in the new cloud services that are appeared after the main Three cloud services namely, Software As a Service(SaaS) , Platform as a Service(Paas), and Infrastructure as a Service(IaaS). In addition to those, they founded: Communication as a Service, Monitoring as a Service, Security as a service, and Database as a Service. Final results that are found, after examining the encryption algorithms: AES128, AES192, AES256, and DESI168, is that AES128 had scorred the most efficiency degree in terms of time consumed to perform such as task, size that it takes from memory while executing, and CPU effort that is spent to execute.

The issue with this paper [8] is that it had not clearly identified the type of files and the access methods. As a suggestion and in order to refine the perfomance, we prefer that analysis of the data files organization should take place firstly, and then decide how to organize the files and how to access data on them , one can use for intance a binary tree to organize the records, and so on depending of which suits what.

The following table summarize further studies that are not discussed in details in the preceding sections in the current chapter, and also short comment are given.

Table 2.2: summarization for related studies

Author	Title	Comment
Prof. S. S. Asole, Ms. S. M. Mundada, 2013	A Survey on Securing Database From Unauthorized Users[9]	- Survey study -refered to the importance of dual security (contents and access) -contents security mechanisms that the paper discussed : a. cryptography, b.steganography, c.dynamic steganography, d.visual cryptography, e.extended visual cryptography

		- Each of which(the sec. contnets) are suitable in some security as well as performance level.
Iqra Bashart, Farooque Azam, and Abdul Wahab Muzaffar, 2012	Databse Security and Encryption: A Survey Study[10]	-A survey study -Had clearly identified 'weak authentication' as one of the major security risks to databases. That leads to obtain in-memory database transaction information illegally. -Had presented many studies that discuss database encryption as such.
D. Sathiya, S. Albert Rabara, and J. Ronald Martin	A framework for Secure Mobile Database Transactions using Cryptographic Co-processor.[11]	-A very useful method that the paper discussed for enhancing transaction security using embedded processor for encrypting/decrypting .
MANDEEP KAUR, and MANISH MAHAJAN	Using encryption Algorithms to enhance the Data Security in Cloud Computing[12]	-It discusses a theoretical formulation of solution to secure cloud data. -AES, Blowfhish, RSA are suggested encryption algorithms to be employed.
D.Roselin Selvarani, and, Dr. T. N. Ravi.	A Review on the role of Encryption in Mobile Database Security[13]	-Discussed three issues should maintain carefully in mobile databases, namely: securing mobile device, mobile database security, and security of

		mobile network. -Discussed that encryption can play big role to secure the channel between Mobile client and the server.

CHAPTERTHREE

Performance Analysis of The algorithms, the methodology

Chapter (3)
Methodology and application

3.1 Introduction

The experiment based on the variables mentioned previously in section 1.6. Each algorithm will be executed in the same machine with different types of database transactions as well as different lengths of data if applicable in such transaction. For each case the experiment of each algorithm will be performed several times then the average will be the final result with such case. Comparisons using statistical charts will clearly identify differences of performances in terms of time, CPU efforts used, and memory consumption for each algorithm used to encrypt each transaction type .

3.2 Algorithms evaluation

Firstly, the samples will be selected and grouped into different groups, each group includes database transactions of a specific type. For the group read-only and read-write transactions, sub-groups (clusters) are going to be formed according to the length of data being encrypted. Each single sample will be encrypted several times given the same environment, results (time consumed to encrypt the transaction each time) will be stored in a database to calculate the average as well as deviations when changing the length of such data.

The time measurement will be calculated by using a stopwatch; two time captures in milliseconds will take place, one will be before invoking the intended algorithm, and the other will be after performing the tasks of that algorithm. By subtracting the second time capture from the first one we can accurately identify the time consumed to perform the encryption of that method.

Actually, the calculated time will include the task of invoking the method, loading the method into the memory and the thirdly, the instructions included in that method.

3.3 The Experiment Environment

The environment in which the experiments were performed are as follows :

- Hardware :

  - ✓ Toshiba C50 Laptop .
  - ✓ Corei3 Intel processor.
  - ✓ Processor speed : 2.40 GHz.
  - ✓ 4 GB RAM.
- Operating System: Microsoft windows7 professional.
- File organization from which the data is read: NTFS.

  Notice that data are put in files for the purpose of reading such data rather than putting it into inner code because of methods and programs sizes limitations. But Actually, when performing the experiment at real situation, data will be read directly from RAM (the main memory).

3.4 Tools and programs :

- Java programming language for algorithms programming.
- Netbeans would be the framework to perform and run processes.
- Microsoft Excell is the demonstration tool to analyze and illustrate results statistically using graphs and notations .

3.5 Data Set:

Data set under the experiment and study are database transactions. In order to evaluate the encryption algorithm in encrypting each transaction, and for simplicity purposes as well as for defining to what extent the length of data affect the evaluation and measuring such performance accurately.

Database transactions are categorized into two main parts in order to perform the experiments of our research on: the part of the transaction which identify the transaction type as well as its id, and hereby the researcher will refer to it by 'transaction header'. The second one is the actual data that is held by the transaction and which is already had been manipulated or read from the database.

The structure of each transaction differs a little bit from others according to the type of any transaction. The following clarify every one of them:

a. For starting a transaction: [*start_transaction,T*]
b. For transaction commitment : [*commit, T*]
c. For abort transactions : [*abort, T*]
d. For read-only transaction : [*read_item,T,X*]
e. For read-write transaction : [*write_item,T,X,old_value,new_value*]

Where: T → the transaction id

X → the data item that is modified.

The second part of transaction is the data part. We divided it into three main categories in terms of length for the research purpose: the first one is a data length of 'VARCHAR2' data type in mysql DBMS, that is of 65535 character; i.e., 65.535 kilobytes. The second category of data consists of 32.768 kilobytes. And The third consists of 16.384 kilobytes.

For all three data categories we've just mentioned in the preceding paragraph, random characters had been taken using the random numbers that are generated randomly as well by the trusted scientific website: 'randomizer.org'[12-11]. Generated random numbers are in the range (33 - 255) and each number generated is casted into the equivalent character; that is to say, each number is considered as the ASCI code of the intended character. As it had been mentioned in just last sentence, the available characters and symbols those are possible are list in the appendix [A]

For each one of the data categories, i.e., 65.553 kilobytes, 32.768 kilobytes, and 16.384 kilobytes, a number of 5 (five) sets are prepared using the method mentioned in the preceding paragraph as they are the random sample of our data that to perform on the experiment. Each set's size will be of size of the data category to which it belongs.

3.6 Determination of time taken for encrypting the transactions:

Generally, each transaction part (either the transaction header or even the data set) is encrypted by the encryption algorithms under study 20 (twenty) times. Then the average value for each part/set is taken as the final result for that algorithm in case of that set/part.

For transaction headers, the value of time taken (in milliseconds) is determined by the following steps:

a. Determine the transaction header that going to be encrypted.
b. Determine the encryption algorithm under study.
c. Perform the encryption process 20 times and store each value (the time consumed to encrypt the transaction header) in a individual database or file.

d. Calculate the time that is taken by algorithm by getting the average of the 20 values that is adopted earlier. The average = $\frac{\sum_{i=1}^{20} time}{20}$.

For Data, determine the size of data first, then select the encryption algorithm under study and then calculated the averages by following method:

a. Determine new set of the given sets that belong to that data category.
b. Perform the encryption process 20 times in each set. Store each value (the time consumed to encrypt that set in millisecond) in a separate database or file.
c. Get the average of those 20 values and store the value, let's refer to it by A_i. The average = $\frac{\sum_{i=1}^{20} time}{20}$.
d. Return to step (a) and perform the steps following it on the other set until you finished the five sets.
e. When finishing calculating five averages, consider the average of those averages, then the calculated value will be considered as the measured time for encrypted data. The average will be calculated as: $\sum_{i=1}^{5} A$.

3.7 Determination of CPU usages by the algorithms under study

The CPU efforts used during the execution of each algorithm under study was captured through the 'Task Manager' tool that is available in Windows7 Operating systems.

3.8 Measuring of memory size consumption by the algorithms under study

Size of memory reserved for running each one of algorithms under study was captured also through the 'Task Manager' tool that is available in Windows7 Operating system

3.9 The encryption algorithms under study:

In this part, a brief description of the way of working for each algorithm under study, namely, the simple substitution, periodic permutation, and Caesar algorithm. Also, the Pseudocode for each one of them will be stated.

3.9.1 Periodic Permutation:

Periodic Permutation is one of the well known classical encryption technique for encrypting text message. The algorithm will calculate the length for each plain text, and then a random permutation of the order of indexes (positions) of each character composing the plain text will take place until all the indexes are shuffled perfectly and completely stored in specific order. Then the index (position) of the first index generated will take place instead of the first character of the original plaintext, then the second one's value will refer to the index of the value that it holds, and so on until the last position.

Pseudocode:

```
P → plain text
L → length of the P
A→ Array of positions
C → cipher text
C ← null
For i=0 to L do
    Generate a random number that is between 0 and L-1
    Ensure that the number is not existing in A , then put the
      number in A[i]
for i to L do
    append the index value of A[i] of the character at
     plaintext to C
```

3.9.2 Simple substitution

The idea of this algorithm is that the DBMS should have a specific array or any other storage palace(file, database, ..etc) to store all possible characters and their substitutes. Each character has another character that will be replaced by it as its cipher .

The following lines illustrates the algorithm of simple substitution :

```
P → Plain text
L → Length of the plain text
```

```
Sub_arr → a golobal two dimensional array that contains the characters and
          their substitutes .
C→ Cipher text
C ← “”
Block A: If   Sub_arr is filled in      then
   from i=0 to L do
      Look for the substitute of the character of index i from the plain text
       Append that substitute to C
else
       Fill in the sub_arr
        Execute Block A
```

3.9.3 Caesar algorithm

Caesar algorithm, one known classical encryption algorithm, substitutes each character of the plain text with the 3rd character that follows each one from a given list contains all characters in a specific order. The key; which is 3 in case of the original algorithm, can be changed several times but both parties – the sender and receiver – should get noticed of that change.

```
P→plain text
L→Length of P
K→ the key
C→ Cipher text
K←3
C←null
for i=0  to  L  do
   C← C+ the K^th character from the characters list from the plain character of [i] index from P
```

CHAPTER FOUR

Results and Discussion

Chapter (4)
Results and Discussion

4.1 Introduction

The final results obtained in this chapter is based on different phases followed to obtain them. Results are translated in both numeric representations in tabular manner as well as statistical graphs to clearly identify the variances in results in comparing the algorithms under study with each others.

4.2 Results and Evaluation of Algorithms in encrypting Transaction Header (the transaction type and its ID parts) in terms of Time

In this section the researcher browse the evaluation and results of processing time that are taken by the algorithms under study: Periodic Permutation, Caesar, and Simple Substitution algorithm. In each subsection a statistical graphs will be available to clarify typically the differences among each algorithm from the others and how each one acts according to the given type of each transaction under processing.

The following table represents average times taken in milliseconds to process each type of transaction assuming that the transaction has an ID of: 1234567891012345678910 (22 digits). The encryption included the transaction type which can either be a commit, an abort, starting a new transaction, read only transaction, or read-write transaction. Then a comma followed by it, then the transaction id comes finally.

Table0.1:Times in milliseconds that are taken to encrypt transaction headers by different encryption algorithms

Type of transaction	Typical encrypted data	Time taken by “Simple substitution” algorithm	Time taken by “Caesar” algorithm	Time taken by “Periodic Permutation” algorithm
Read only transaction	*read_item,1234567891012345678910*	0.0879628	0.02069715	0.0456698

Start transaction	*start_transaction,12345678910123456 78910*	0.09786245	0.01610025	0.0593548
Commit	*commit,12345678910123456 78910*	0.0748133	0.0123584	0.048236199
Abort	*abort,12345678910123456 78910*	0.05255535	0.0173404	0.0558486
Read-write transaction	*write_item,12345678910123456 78910, database.emp.name*	0.08625235	0.0558486	0.0974134

Obviously, Caesar Algorithm had recorded the minimum time taken to encrypt the 5 types of transaction headers, then Periodic Permutation algorithm recorded the second score, then lastly the simple substitution set place. Figure 3.1 illustrates the differences in times taken by each algorithm in encrypting each type of transaction header.

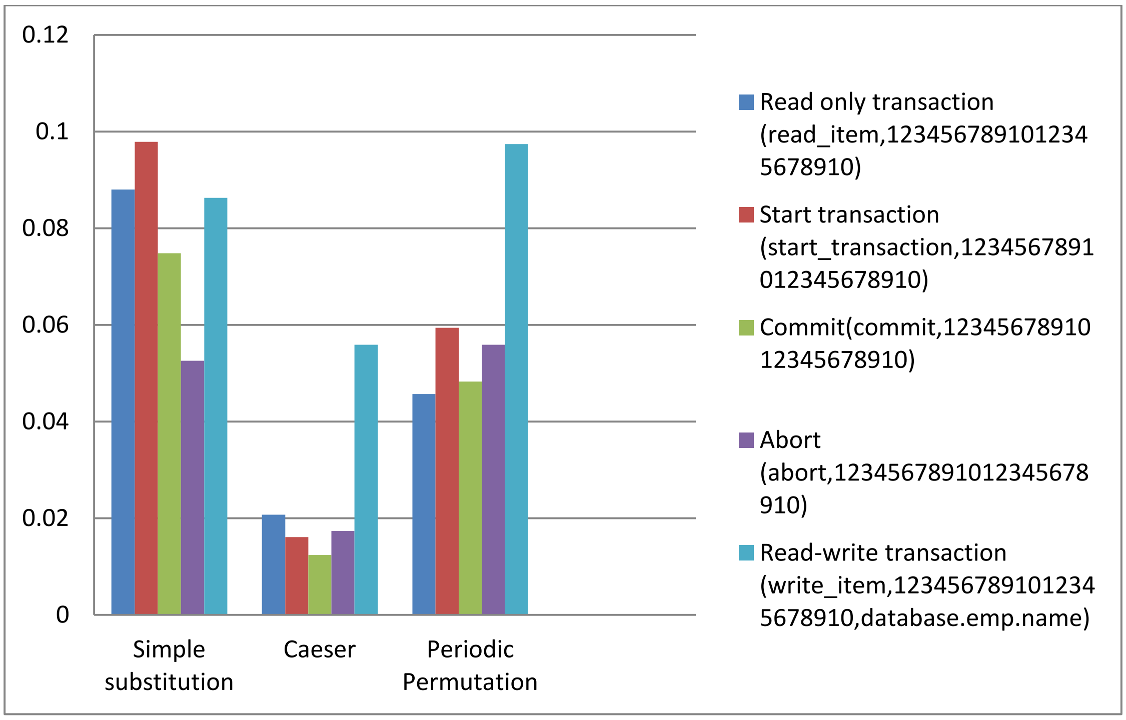

Figure 0.1 : Time spent in milliseconds for encrypting different database transaction headers types

In another hand, it's clear that type "started transaction" has recorded the biggest amount of time it took to encrypt by both simple substitution and, Periodic Permutation algorithms. Whereas by Caesar, it took less time than the read-only type did, but more time than the commit type did take.

Also, it is clearly observed that the read-write transaction in all three algorithms consumes more time than the other transaction headers' types. While the start transaction encryption by the simple substitution algorithm recorded the biggest amount in all experiments from other types, Time consumed by commit transaction by Caesar algorithm had recorded the most little time than all other transaction headers that are encrypted by the same algorithm as well as by other algorithms.

Finally, according to both table 3.1 and figure3.1 above, Caesar had won the competition than other encryption algorithm, and we can ensure that it is the optimal one for encrypting such transaction header data; that is to say it is the most suitable option to be considered in terms of time in encrypting the first part of the transaction header (the part that consists of the transaction type and its id). But for Security issues, a regular change of the key each must be done several times to harden the process of cryptanalysis that can be performed by black hat hackers(crackers).

4.3 Evaluation of algorithms in encrypting Data part in the transaction record in terms of time

Data that is considered to obtain our results are categorized into three categories in terms of length. As we referred to that in the preceding chapter, 1st category is of size 65535 which is typically 65.535 kilobytes which is the maximum number of characters that the Varchar2 data type has in mysql DBMS. The 2nd one is of the half of the maximum size of varchar2 which is 32768 characters. The 3rd category is of 16.384 kilobytes.

Table 4.2 : Times taken for encrypting different data sizes using encryption algorithms mentioned below.

	65.535 kilobytes	**32.768 kilobytes**	**16.384 kilobytes**
Simple substitution	3434.94	862.12	217.16
Caesar	3337.13	843.87	213.49
Periodic Permutation	3340.82	829.21	210.03

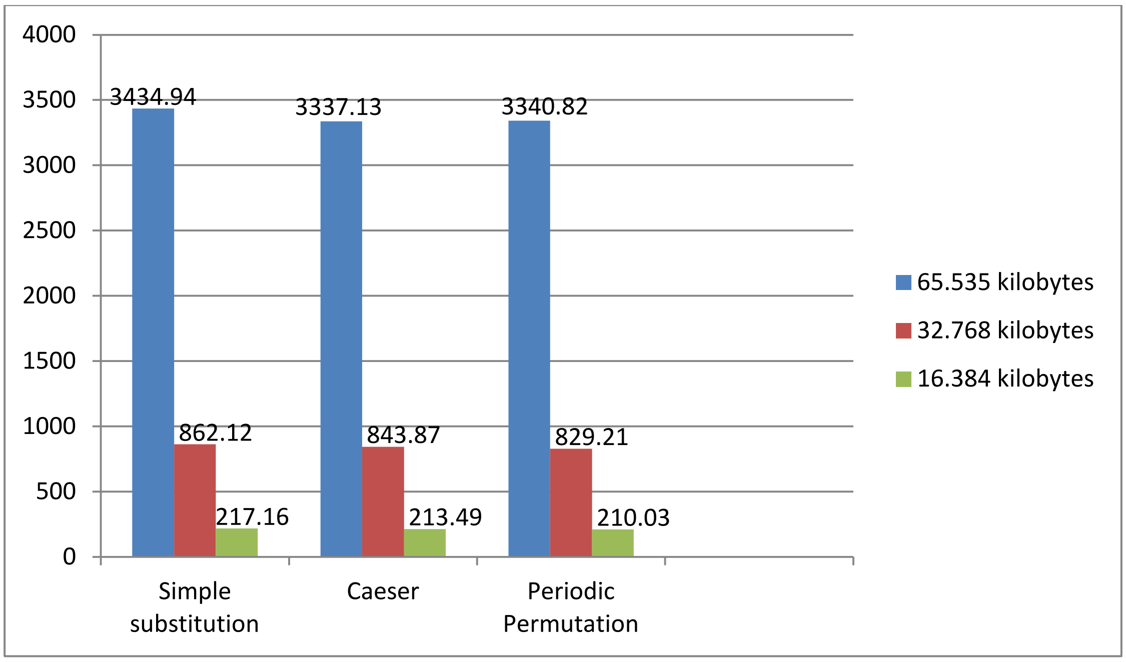

Figure 0.2 : Times spent in milliseconds to encrypt data part by different encryption algorithms

As it had appeared above in Figure 3.3, the three algorithms under study have nearly the same results. Most efforts and time spent to encrypt the data cluster of 65.535 kilobytes was recorded by the Simple Substitution method, It took3434.94 Milliseconds. While the least time spent for encrypting such cluster was 3337.13 Milliseconds by Caesar algorithm.

On the other hand, it has been found that data of 32.768 kilobytes took the largest period of time to be processed was recorded by Simple Substitution algorithm, and least time that the same size of data spent to be encrypted was recorded by Periodic Permutation algorithm.

The last data group with size 16.384 had been processed in nearly the same time via the three of algorithms. But accurately, Periodic Permutation consumed the least time from other with 210.03 milliseconds, while simple substitution had took the biggest plenty of time among the three with value of 217.16. Then, Caesar had got the second score with an average time of 213.49 millisecond consumed for encrypting size of data. The variance between time spent in processing by Simple Substitutions and Caesar was just 4 milliseconds, and between Caesar & Periodic Permutation was only 3 millisecond, this is an indicator of stability in data processing with mentioned size, so deciding which to choose cannot be easy, therefore, including other methods to support decision making such as machine learning will be valuable for more accurate decisions .

4.4 Results and Evaluation of performance of encrypting the 'data' transaction section in terms of CPU efforts usage

After examining the pre-mentioned categories of data; the category of 65.535 kilobyte, 32.768, and of 16.384 kilobytes. The results that the researcher arrived at is as follows :

Table 4.3 : Effort of CPU (in percentage) taken for encrypting different data sizes using encryption algorithms mentioned below.

	65535 kilobytes of data	**32768 kilobytes of data**	**16384 kilobytes of data**
Simple Substitution	19.46714286 %	14.46%	0.5%
Periodic Permutation	23.964%	24.03%	20.725%
Caeser	25.02%	23.84%	20.825%

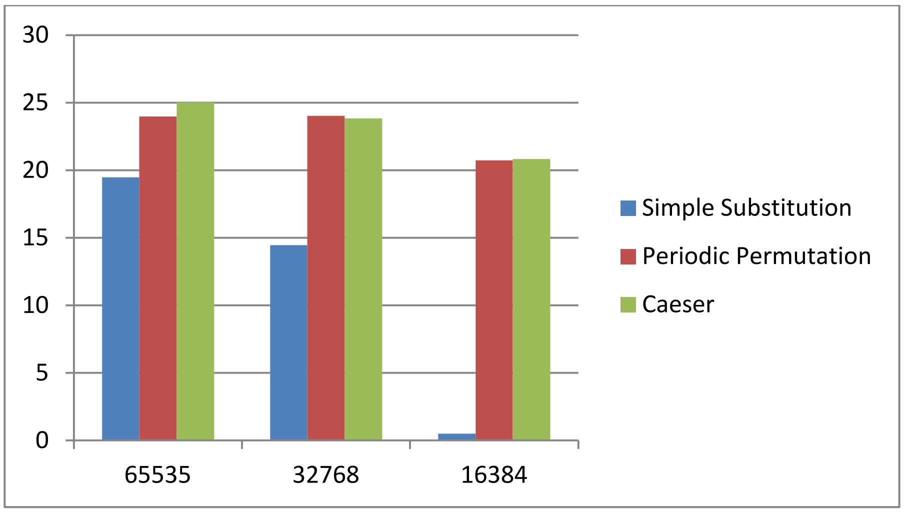

Figure 0.3 :CPU effort consumed (in percentage)for encrypting data part by different encryption algorithms

From the Chart above, the reader can see that Caeser algorithm had scored the last position in the competition between it and the two other algorithms in encrypting the largest size of data under study – 65.535 kilobytes of data . Whereas Simple Substitution algorithm was the optimal one in encrypting such size of data.

Simple Substitution algorithm also came at the first place when encrypting the data of size 32.768 kilobytes, and regarding to the other two, the CPU efforts used for both was nearly the same but ‘Periodic Permutation’ had consumed a little bit more efforts than that Caeser did.

Regarding the third category of data –size 16.384 kilobytes – Simple Substitution algorithm won the competition once again with a very big difference in efforts that it did require from the CPU to perform comparing to the other two, it required just 0.5 of the CPU efforts. And once again, the other remaining algorithms had nearly a similar percentage of CPU efforts consumption but Caeser algorithm this time came at the tail of the list.

4.5 Results and Evaluation of performance of encrypting ‘data’ part of transaction in terms of Memory reserved for doing so:

‘Task Manager’ tools is also used for measuring the memory usage rate in kilo bytes for each algorithm to perform. Following table shows size of memory reserved and used for encrypting our data.

Table 4.4 : Sizes of memory reserved for encrypting different data sizes using encryption algorithms mentioned below.

	65.535 kilobytes of data	**32.768 kilobytes of data**	**16.384 kilobytes of data**
Simple substitution	353153.3333	293313.5333	18367.54545
Caeser	234924	351613.3333	353001.3333
Periodic Permutation	298140	295021.3333	295746

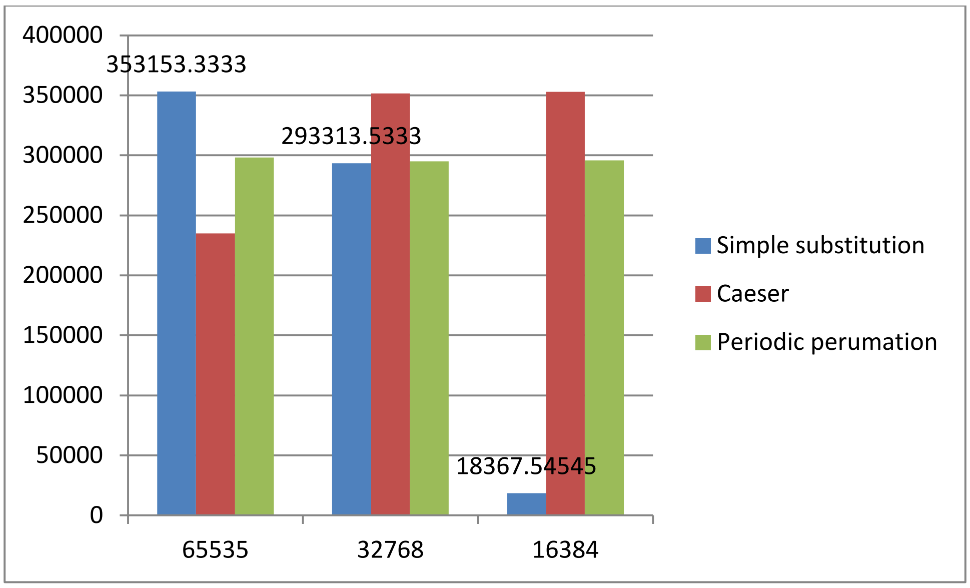

Figure 0.4: Memory sizes reserved for encrypting data part by different encryption algorithms

As it is shown above, it’s clearly that Caeser algorithm had won the competition and registered the first place in terms of memory consumption in large data – of size 65535 kilobytes . Whereas Simple substitution reserved the smallest memory size when encrypting 32.768 kilobytes of data while Caeser consumed the largest size of memory comparing to the other two, so it came at the tail – the third place.

Regarding data of size : 16.384 , Simple Substitution came also at the first place while Caeser had lost the competition once again and took extremely large size of memory in order to perform.

4.6 Discussion and Gap filled

We’ve introduced the performance of algorithms under study in terms of time, CPU efforts, and in terms of Memory usage. Also, we had performed a quick comparison between them in processing different types of transaction and also the different sizes of data and how do they affect the processing time. Results were stated in both tabular representation and statistical charts.

Results show that Caesar algorithm was the optimal one for encrypting small-sized data but it took too much memory size for running . On the other hand, the same algorithm had scored an average performance, it took more time than Simple Substitution algorithm, but less than Periodic Permutation did take to encrypt data of size range 32.768 – 16.384 kilobytes.

In the case of data sized 65.535 kilobytes Caesar had recorded the first place in the competition in terms of how fast the algorithm will work, then periodic permutation got the second place, and lastly simple substitution algorithm came with extremely big difference and variance of time between it and of simple substituion’s. In terms of CPU efforts usage of such size of data, Caeser consumed the most efforts of CPU than did the other take while Simple Substitution algorithm had won the competition in this case. Memory-wise, Caeser was the most economic algorithm when encrypting this large data, Periodic Permutation came at the 2nd place, then Simple Substitution was the worst in consuming memory area.

As we’re made use of encryption for securing database transaction, we should refer to that periodic permutation algorithm used for encrypting large data usually, so encrypting transaction header doesn’t make sense since the probabilities of decipher the cipher text correctly through cryptanalysis is too easy, that is because the letters used for specifying types of transaction are known and limited. But for research purpose we performed it for completing the big image of the algorithms measurement and comparing it with others.

All in all, taking in consideration the types of files and their access method is not of our scope here in since the data will be read directly from memory; from the cache buffer in order to encrypt via the encryption algorithm. But that issue was too important in case of the paper titled : “performance Analysis of the Encryption Algorithms as Solution to Cloud Database Security” that we referred to in section 2.10 above.

4.7 Chapter summary

This chapter describes the analysis based on the methods of evaluation we introduced earlier in addition to the results we obtained. Also, a brief discussion about the results and our comments on the results had been stated.

CHAPTER FIVE

Conclusion and Future work

Chapter (5)
Conclusion and Future work

5.1 Conclusion

In chapter one we introduced brief description of our research background, motivation, research questions and objectives.

In chapter two we gave an overview of literature review. We got deeply into scientific background of the methodology we specified earlier the in first chapter. Many studies had been discussed in such chapter.

Next chapter – chapter three – we went in details into the methodology which it had introduced earlier in chapter one. It is where we described how the experiment will perform in order to lead us to final results we seek to.

In chapter 4, we had described the results obtained after the experiment had performed. Also, we gave a brief discussion about how our research is participating in supporting actual issues regarding database transactions security.

5.2 Future work

As we mentioned in early pages in this research in chapter 1, the motivation for this research was to enhance the security of in-memory database transactions. So, this work can be one of the basics that can be considered in order to enable database designers make decisions on what algorithms are more adequate in terms of time, CPU efforts, and memory size they take.

Future work can briefly identify and recommend next points :

- Other classical encryption algorithms should take part in such performance analysis and comparisons.
- Measuring the performance of non-text data should take place using special algorithms for doing so.
- Usually, Systems differ in capabilities in processing equipments, i.e., CPU, Cache, RAM, …etc. And for further security guarantees a DBMS can use more than one encryption algorithm, one at a time. Therefore, introducing machine

learning techniques and other decision support systems for making such decision can benefit a lot, but this needs strong and high performance of the processing units.

- Using non-traditional encryption such as 3DES, AES, and so on to encrypt such transactions and to analyze their performance should also take chance to do so that to complete the big picture of complete performance analysis for encrypting database transactions.

REFERENCES

[1] Dorothy Elizabeth Robling Denning, "Cryptography and Data Security",ISNB: 0-201-10150-5, Addison-Wesely publishing company,1982.

[2] Ron Ben Natan, "Implementing Database Security and Auditing", ISBN:1-55558-334-2, Elsevier Digital Press, 2005

[3] Thomas Connolly, Carolyn Begg, “Database Systems: A Practical Approach to Design, Implementation and Management”, 3rd ed., Dorling Kindersley (India) Pvt. Ltd., New Delhi, 2002.

[4] Reena Gangwar, M.K. Sharma, "Database Security Measurements Issues in Adhoc Network", International Conference of Advance Research and Innovation, 2015.

[5] Ramez Elmasri and Shamkant B. Navathe, "Fundementals of Database Systems", 6th ed.,ISBN:978-0-136-08620-8, Library of Congress Calatolgin-in-Publication Data,2016.

[6] D. Roselin Selvarani , and Dr. T. N. Ravi, "A survey on Data and Transaction Management in Mobile Databases", International Journal of Databas Management Systems (IJDMS), Oct, 2012.

[7] Weider Yu, D., Tamseela Amjad, Himani Goel and Tanakom Talawat, “An Approach of Mobile Database Design methodology for Mobile Software Solutions”, The 3rd International Conference on Grid and Pervasive Computing - Workshops, IEEE 2008.

[8] Murtala Aminu Baba et al…,"Performance Analysis of the Encrytpion Algorithms as Solution to Cloud Database Security" Evaluation of Mobile Agent Platforms", International Journal of Computer Applications, Volume99, August, 2014.

[9] Prof. S. S. Asole, Ms. S. M. Mundada,"A survey on Securing Databases From Unauthorized Users", International Journal of Science & Technology Research, -ISSN: 2277-8616, Aprile,2013

[10] Iqra Basharat , and etal, "Databse Security and Encryption: A Survey Study", International Journal of Computer Applications, volume 47 No.12, June, 2012.

[11] D. Sathiya, etal, "A framework for Secure Mobile Database Transactions using Cryptographic Co-processor", International Journal of Computer Applications, Volume 105 No. 5, November, 2014.

[12] Mandeep Kaur, and Manish Mahajan, "Using encryption Algorithms to enhance the Data Security in Cloud Computing", International Journal of Communication and Computer Technologies, Volume01 No.12, January, 2013.

[13] D. Roselin Selvarni, and, Dr. T. N. Ravi, "A Review on the role of Encryption in Mobile Database Security", Internation Journal of Application or Innovation in Engineering and Management, 2014.

[14] Parviz Ghorbanzadeh, Aytak shaddeli, Roghieh Malekzadeh, Zoleikha Jahanbakhsh, "*A Survey of Mobile Database Security Threats and Solutions for It*", 3rd International Conference on Information Sciences and Interaction Sciences (ICIS), 676-682, China, 2010

[15] Gaurav Sharma, Ajay Kakkar " Cryptography Algorithms and approaches used for data security" International Journal of Scientific & Engineering Research Volume 3, Issue 6, June-2012 1 ISSN 2229-5518.

[16] Urbaniak, G. C., & Plous, S. (2013). Research Randomizer (Version 4.0) [Computer software]. Retrieved on January 12, 2016, from http://www.randomizer.org/

Appendix (A)

List of Possible Characters considered in the experiment

Characters are starting from whose ASCI code is 33 up to 255

!	"	#	$	%	&	'	(	)	*
+	,	-	.	/	0	1	2	3	4
5	6	7	8	9	:	;	<	=	>
?	@	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R
S	T	U	V	W	X	Y	Z	[	\
]	^	_	`	A	b	c	d	e	f
g	h	i	j	K	l	m	n	o	p
q	r	s	t	U	v	w	x	y	z
{	|	}	~	€	Š	‹	Œ	‘	’
“	”	•	–	—	˜	™	š	›	œ
ž	Ÿ	¡	¢	£	¤	¥	¦	§	¨
©	ª	«	¬	-	°	±	²	³	´
µ	¶	·	¸	¹	º	»	¼	½	¾
¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È
É	Ê	Ë	Ì	Í	Î	Ï	Ð	Ñ	Ò
Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü
Ý	Þ	ß	à	á	â	ã	ä	å	æ
ç	è	é	ê	ë	ì	í	î	ï	ð
ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú
û	ü	ý	þ	Ÿ					