DEGREE PROJECT

Bench-marking Framework for Transparent Data Encryption Systems

Feidias Moulianitakis
Konstantinos Asimakopoulos

Information Security, master's level (120 credits)
2019

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Benchmarking Framework for Transparent Data Encryption Systems

Authors: Feidias Moulianitakis
Konstantinos Asimakopoulos

Supervisor: Martin Lundgren

Abstract

In the digital world of today, information is always at risk regardless of its state, at rest or in transit. Cryptography is the technology that promises to address the security issues that emerge. Hence, it was a reasonable consequence to introduce cryptography to databases. However, manually encrypting and decrypting data along with the key management is a burden for the regular user of a database. The need for removing this burden gave birth to Transparent Data Encryption (TDE).

TDE technology is widely available nowadays and a number of vendors have developed their own solutions for protecting data at rest in a transparent way to the end user. However, cryptographic operations are resource intensive and introduce an overhead to the computational operations. The burden of cryptographic operations has drawn the interest of both academia and the industry for a long time before TDE appeared on the horizon. Hence, a lot of research has been done to measure the performance impact of those operations.

Despite the extensive study for the performance of cryptographic algorithms, the performance of the TDE systems and the add-on computational burden for the introduced encryption has not yet been studied thoroughly. As a result, the current Thesis project tries to develop a theoretical benchmarking framework that evaluates the performance of Transparent Data Encryption systems. The study is conducted utilizing the Design Research methodology.

The developed benchmarking framework focuses on the basic performance metrics of TDE systems, Elapsed time, CPU time and Hard Disk memory consumption. These metrics are calculated for varying key lengths, encryption algorithms and table sizes. The framework follows a five - step procedure that includes the creation of topology - lab environment, creation of databases and definition of scenarios, activation of TDE feature, sequencial execution of scenarios and analysis of the results. The developed framework is evaluated by applying it on real TDE systems.

Keywords: Transparent Data Encryption, benchmarking framework, encryption performance, Design Research, Cryptography.

Table of Contents

List of acronyms and abbreviations

3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
API	Application Programming Interface
CPU	Central Processing Unit
DB	Database
DEK	Database Encryption Key
DES	Data Encryption Standard
DPAPI	Data Protection API
DRM	Design Research Methodology
GIS	Geographical Information Systems
IT	Information Technology
MB	Megabyte
ms	Microseconds
NIST	US National Institute of Standards and Technology
RC2/RC6	Rivest Cipher (2/6)
s	Seconds
SQL	Structured Query Language
TDE	Transparent Data Encryption
VM	Virtual Machine
VTE	Vormetric Transparent Encryption Solution

List of Figures

List of Tables

1. Introduction

This document is intended to develop a theoretical framework that will be utilized to benchmark Transparent Data Encryption (TDE) systems. The current Chapter, explores the TDE concept and how this technology is of crucial importance nowadays. This is achieved by briefly presenting the related literature which leads to the identification of the research gap that the current research project attempts to address. Finally, the contribution of the present research along with the imposed delimitations are briefly discussed.

1.1 Background

The world is eternally changing, evolving and now has already entered a new era, the era of the Fourth Industrial Revolution [1]. In this new era, Information Technology (IT) does not only play a role in industrial production but it penetrates people's daily lives and gradually becomes inseparable tool for work as well as personal life [2] [3]. The generated data need to be stored and organized in order to be always available to its users. Websites with audio/video files, online telephone catalogs, Geographical Information Systems (GIS) are collections of data that people use daily. Such collections of related data are called databases [4].

Digitizing all this information, produces multiple security risks. Malicious individuals exploit software vulnerabilities in order to gain unauthorised access to systems. In this new digital world personal data need to be secured regardless of its status, at rest or in transit. This can be achieved by utilizing the science of Cryptography. One solution is to encrypt the whole disk, where the database is stored, compromising with the fact that the whole disk needs to be decrypted in order to access the stored data. Management of the encryption keys is an additional burden for such applications. Another solution is to handover the security of the data to a third party and store it on the Cloud but in this way, security becomes a black box and there is always the risk that the provided security does not meet the desired level.

A third solution could be to secure the data at the client side before it is stored to the database by utilizing end-to-end encryption applications. This is a promising solution where the key management is a client's responsibility. However, key management is always a burden for the user of such systems. Hence an all inclusive solution to the end-to-end encryption problem along with managing the required encryption keys is provided by the Transparent Data Encryption (TDE) technology [5] [6].

TDE is used to encrypt data-at-rest in database environments. The term data-at-rest represents data that are stored, in any digital form, in a physical location including cloud services. TDE works at the file level of a database and depending on the specific software it could affect further down at the column level. It was introduced with Microsoft SQL Server 2008 [5], and now is offered by other major vendors as well [7] [8]. The term transparent comes from the fact that the mechanism behind the encryption is not visible or understandable to the end user (human or process). Technically, the end user does not provide any encryption/decryption key

or password in order to retrieve the data. The process is handled by the relevant software. This TDE software is bound to decrypt the requested data, before fetching them to the user and encrypt the data before they are stored back to the database.

1.2 Related work and research gap

As previously discussed, we are living in the era of information. Data is transmitted and stored in various means which increases the surface of attack for potential hackers. Thus, protecting the information data is of crucial importance. The significance of the field has drawn the attention of both academia and industry. In this section a number of research papers in the field are briefly described.

In his paper, Sourav Mukherjee, highlights that in today's world data is more valuable than the expensive hardware where it is stored and must be carefully protected [9]. This can be achieved with cryptography. Hence, the databases where data are stored and organized must be encrypted and according to Sourav Mukherjee there are four encryption models to achieve this, namely Always Encrypted, TDE and Cell Level Encryption, Dynamic Data Masking and Vormetric Transparent Encryption (VTE). The paper concludes that VTE is the best model since it protects data across its whole lifecycle including when it is backed up, migrated and archived as well as it provides transparent encryption and access control to the file, folder or volume level.

Deshmukh and Qureshi, with their paper, analyse the term TDE [10]. They state that it is the technology that promises to solve the problem of data security. TDE provides transparent to the user (human or process), standards-based security to the data on the network, on disk or on backup media. Data and consequently databases where they are contained are vulnerable to a number of threats such as unauthorized access, week authentication, backup exposure, privilege abuse etc. TDE can be utilized to prevent unauthorized access to confidential databases, reduce the cost of managing users, and facilitate privacy management as well as meet regulatory compliance or corporate data security standards. These are achieved with encryption of data before storing it in the database and decryption of it before returning it to the application in a completely transparent way to both the application and the user of it.

Cryptography is indeed the solution to the problem of securing confidential data in databases. However, the utilization of encryption algorithms is expected to degrade the performance of the system since it requires computationally intensive mathematical calculations. Elminaam et al, with their paper, propose a model to evaluate the performance of six of the most common symmetric encryption algorithms namely: AES, DES, 3DES, RC2, Blowfish and RC6 [11]. Asymmetric encryption algorithms have been excluded from the evaluation since they are almost 1000 times slower than symmetric algorithms and they require even more processing power [12]. The experimental evaluation has been conducted with the following metrics: first is the encryption time which is the actual time to produce the cipher text from the plaintext, second is the CPU processing time which is the time that the CPU is committed only to the encryption/decryption task and finally the CPU clock cycles which is a metric to calculate

energy consumption. These metrics have been calculated for various scenarios such as change in the data type (e.g text, document or image), changing packet size and changing key size.

Another interesting comparison of encryption algorithms has been performed in the paper of Abdel-Karim Al Tamimi et al [13]. The ability to protect data and on the other hand speed and efficiency are the two main characteristics that differentiate the encryption algorithms. The paper compares four algorithms namely DES, 3DES, Blowfish and AES (Rijndael). The implementation is based in .NET environment and uses its libraries. In this paper the factor to evaluate the performance is the "speed to encrypt/decrypt data blocks of various sizes".

Finally, Hu and Klein in their paper try to combine the above discussions and benchmark the TDE for migrated databases to the cloud [14]. They suggest that the most efficient way to protect data in the cloud and in parallel provide Transparent Encryption Services to the end user is by inserting to the system a middleware encryption layer between the front-end application and the back-end databases. This middleware layer, according to the authors, is a fine-grained, transparent and vendor independent solution for protecting data stored in the cloud. The benchmarking has resulted in no significant impact on the application startup, encryption increases the time cost of operations and the space cost is determined by the encryption algorithms utilized.

1.3 Contribution and research questions

As discussed in the previous sections, TDE is a promising solution for the protection of data at rest stored in databases. By the time that the current Thesis project is conducted and according to the best of our knowledge, the academic research has mainly focused on the performance of encryption algorithms and databases. On the contrary, the performance of the TDE systems and the add-on computational burden for the introduced encryption by the TDE technology has not yet been studied thoroughly. Consequently, the goal and contribution of the current Thesis is to develop a benchmarking framework that evaluates the performance of TDE systems. This benchmarking framework aims to highlight the additional computational burden introduced by the encryption involved, validate its "transparency" feature and can be utilized by companies or decision makers for investing in such systems.

According to the discussion on the problem area and the research gap identified above, the main research question that the current Thesis project attempts to answer is the following:

- How to methodically evaluate the performance of Transparent Data Encryption systems?

1.4 Delimitations

TDE is a technology with a number of configurations and use cases. Due to the limited time that has been allocated for the current Thesis project, a number of delimitations needs to be introduced. The most significant of them is that the developed framework focuses only on the

performance of TDE systems and the additional computational burden introduced. However, security is a crucial attribute of such systems and its strength along with potential vulnerabilities of the TDE implementations are an important topic that needs further investigation. Despite of this, the security strength of the TDE systems is not in the scope of the current project and is left as future work.

The developed benchmarking framework is validated by testing it on two databases that support TDE. This fact can provide a proof of concept. It would be interesting to perform a comparison of a more extended number of TDE systems that are available on the market. However, this could not be performed as part of this Thesis project due to the limited time and available resources.

Finally, the last delimitation is the deployment. The scenarios executed for evaluating the developed benchmarking framework have been conducted on VMs. Due to the available resources, it was not possible to create a testbed with multiple physical servers in order to simulate real enterprise scenarios.

1.5 Structure of Thesis

The current chapter was an introduction to the research area of this Thesis work by presenting the conducted literature review, the identification of the research gap, motivation and contribution of the Thesis project as well as a brief discussion of the project's delimitations. The rest of this Thesis report is structured as follows: Chapter 2 describes the basic theoretical background required for the analysis. Chapter 3 presents the research methodology utilized for answering the research question defined in Section 1.3. Chapter 4 introduces the proposed benchmarking framework which is later applied and tested against real TDE systems in Chapter 5. Chapter 6 includes a discussion of the benchmarking framework and its evaluation as well as it realizes some further work that could be done in the field. Finally, Chapter 7 concludes the Thesis report by providing answers to the addressed research question.

2. Theory

In the current Section, it is attempted to develop some background knowledge and provide an insight into the field of study. Hence, the TDE technology and its architecture is briefly discussed along with the basic metrics used to evaluate the performance of a TDE system. Finally, the Section ends with a brief discussion of the encryption mechanisms utilized by the TDE systems.

2.1 Transparent Data Encryption in a few words

TDE is a technology born for protecting the data stored in databases. The idea behind it, which also explains the keyword “Transparent”, is to secure the data that are stored without the user being aware of the security processes involved [5][6]. Hence, in order to achieve the protection of the data, they need to be encrypted while they rest in the database. Furthermore, in order to make the whole procedure transparent to the user of the data, the encryption keys must be managed by the TDE system. Last but not least, the TDE system needs to perform real time I/O encryption of data as well as of database log files otherwise the procedure would be noticeable from the user.

As it has already been discussed, it was initially developed by Microsoft for Microsoft SQL Server 2008 [5]. The architecture of TDE according to Microsoft is illustrated in Figure 2.1 [15]. In this figure, it can be observed that the architecture starts from the Windows Operating System level with the use of the Data Protection API (DPAPI) [16]. DPAPI is responsible to encrypt the Service Master Key that is generated at the time of SQL Server setup. This key is unique per SQL Server and is utilized to encrypt the Database Master Key for the master database. In turn the Database Master Key is utilized to create a Certificate [17]. This Certificate is used to encrypt the Database Encryption Key (DEK) which is the actual key used by the TDE system to encrypt/decrypt the user’s database.

Protecting the DEK with a certificate is important because if someone manages to extract the encrypted database from the system and tries to read it in another location, without the keys protected by the certificate he or she will not be able to extract the actual data. Finally, data are protected from the TDE while at rest. While in transit or in memory, the data are decrypted by the TDE system and at that stage protection must be provided by other means.

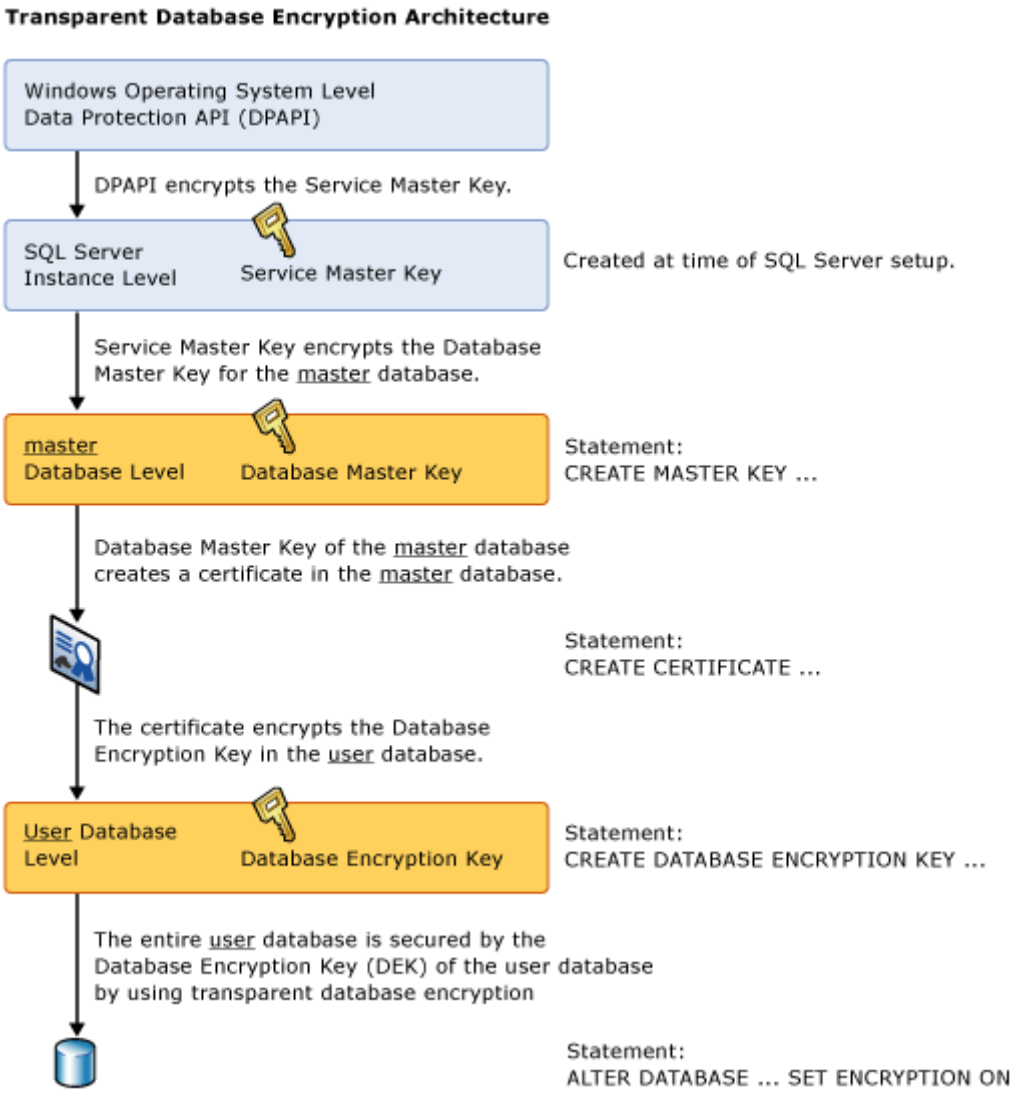

Figure 2. 1: Transparent Database Encryption Architecture [16]

2.2 Performance Metrics

TDE is an add-on service to a database. Hence, applying TDE will impose a performance overhead to the actual service. The most common performance metric that can be used to measure this overhead is time. For the scope of this thesis project, time is divided into two distinct categories namely, processing time or CPU time and Elapsed time.

CPU time is the time a process consumes CPU resources, or in other words, the time that the CPU spends on executing the commands of a specific process [18] [19]. This metric excludes I/O operations as well as executions of other processes that are scheduled in parallel with the process measured. The motivation behind this metric is that it indicates the actual computational burden introduced to the process of the database when introducing TDE.

On the other hand, elapsed time is more general and provides values closer to the actual time experienced by the user. Hence, elapsed time can be affected by other processes running on the system at the time of execution as well as I/O operations [20]. This metric is an indication how transparent will eventually be a TDE system since its impact is visible to the user.

Additionally, one more metric can be derived from the Elapsed time, namely Throughput [21]. Throughput, is calculated in MB per second by dividing the size of the input/output of a query with the elapsed time of its execution. Throughput metric highlights the efficiency of the database and thus the TDE impact to its performance. The formula of the derived metric can be found below [22].

$$Throughput = \frac{bytes}{ElapsedTime \times 1024^2} \quad (1)$$

2.3 Cryptographic algorithms

Encryption of data is the key function of every TDE system. Hence, TDEs of all vendors rely on a subset of well-known encryption algorithms with proven cryptographic strength. Two of the most common cryptographic algorithms are the Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES).

AES is the result of the AES selection process that was conducted by the US National Institute of Standards and Technology (NIST) in 2001 [22] [23]. The algorithm was originally known as Rijndael which was the name of the proposal submitted to the selection process by its developers Vincent Rijmen and Joan Daemen [24]. AES is a symmetric-key block cipher with a key length spanning from 128 bits to 256 bits.

On the other hand, 3DES is another well known and widely used symmetric-key block cipher. As its name indicates, the algorithm consists of 3 executions of the DES algorithm (encryption-decryption-encryption). The algorithm was introduced with the RFC 1851 in 1995 [25]. 3DES has more limitations regarding the key length compared to AES due to backwards compatibility with DES. Hence, the key length of 3DES is 192-bits but the actual key-space is 168 bits (the rest of the 24 bits are used for error detection) [26].

3. Method

The current Master Thesis project aims to provide a procedure of conducting a benchmarking analysis of the performance of TDE systems. Subsequently, a secondary goal is the evaluation of the developed benchmarking framework using existing TDE systems. In order to fulfill this goal, Design Research Methodology (DRM) is considered the most appropriate method [27] [28]. DRM is used when there is a need to design an object that solves an identified problem.

The procedure of the Master Thesis project will follow the listed phases:

- Literature study of the different techniques and systems.
- Development of a theoretical framework to benchmark TDE systems.
- Evaluation of the developed benchmarking framework using existing TDE systems.

The first phase, the literature study, consists of studying and reviewing material in the field of the project such as white papers, conference presentations, articles, reports and other online documents. Performing this step will provide the required knowledge over the field of TDE as well as understand the actors involved and the technologies utilized.

In order to conduct the literature review, the following procedure was performed.

1. Search for documentation explaining TDE technology. This provides an overview of the field of study.
2. Search for vendors who develop such systems and investigate their products.
3. Search for research papers investigating the performance of TDE. In this step, the benchmarking of encryption algorithms was investigated as well.
4. The reviewed material was summarized and the relevant points to the scope of the current thesis project were taken into consideration for the following phases of methodology.

The second phase, the development of the theoretical framework, is conducted following the DRM. According to Peffers et al [28], DRM consists of six steps that are not purely sequential. Iterations between the steps may be required in case a step is not successfully fulfilled. The six steps are presented in Figure 3.1.

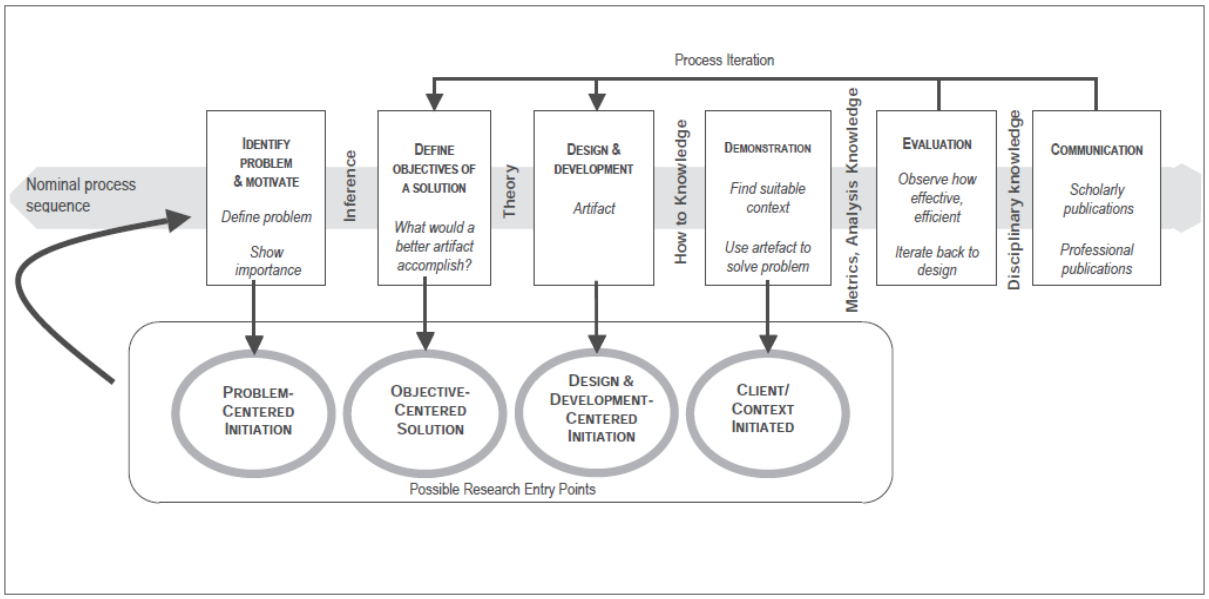

Figure 3. 1: Design Research steps [28]

The first step, **"Identify Problem & Motivate"**, in the procedure presented in Figure 3.1 concerns the identification and motivation of the problem which for the current research is to develop a theoretical framework and evaluate it on existing TDE systems. The scientific research revealed that the encryption algorithms have been examined, compared and benchmarked thoroughly. However, as discussed in the Introduction Section, TDE systems that are crucial for the protection of data at rest have not been examined and benchmarked methodically. Thus, this Thesis project aims to develop a theoretical benchmarking framework in order to evaluate the performance of TDE systems.

The second step of the process, **"Define Objectives of a Solution",** includes the definition of the scope of the proposed solution. In this step, for the current research project, the objectives of the theoretical benchmarking framework are defined and analysed.

In the third step, **"Design & Development",** the Artifact, which in this case is the theoretical benchmarking framework, is actually designed. This step includes the procedure of how to methodically measure the performance of TDE systems. Additionally, the metrics utilized along with a number of parameters such as encryption algorithms, key length and table size are analysed.

Subsequently, the fourth step of the DRM, **"Demonstration",** involves the execution of the benchmarking framework for evaluating and comparing the performance of different TDE systems. For the scope of the current Thesis project, two TDE systems are compared from a performance perspective using the developed framework.

The third phase of the Thesis project starts with step five of the DRM, **"Evaluation"** of the developed theoretical benchmarking framework by conducting it on real TDE systems available on the market. A quantitative approach is used in order to measure and compare the

examined TDE systems. This step might require a significant number of iterations for calibrating the method.

The final step, "**Communication"**, is highlighted by both Peffers et al [28] and Hevner et al [29]. Step six is important since the product of the process, the Artifact, is communicated to the stakeholders as well as to any interested party. It is actually a marketing process that will also enable further improvement via the feedback received.

The motivation behind the choice of DRM as the main method of conduction the present Master Thesis project is based on the fact that the core product of the project is the theoretical benchmarking framework. The framework is the artifact of the DRM. DRM provides a step by step method for designing an artifact and subsequently evaluate its validity which also covers the secondary goal that was set in Section 1.3. Furthermore, DRM is an iterative process that provides the opportunity to review previous steps and, in this way, calibrate the artifact under development in order to improve the final outcome.

The above reasons lead to the choice of DRM as the most appropriate methodology for the problem that the current Master Thesis project attempts to address. In the process of selecting a suitable method, the Action Research Methodology was also considered [29]. This methodology was rejected as non applicable in the specific case. The proposed problem does not affect the operation of a company in which case the Action Research Methodology would be more suitable. Additionally, Action Design Research Methodology is even more interventional in a company's live operation [30]. In addition, it requires many teams to collaborate to the final output. As a result, Action Design Research Methodology was rejected as well. Hence, for these reasons the Design Research Methodology is considered the appropriate methodology for the current Master Thesis project.

The Thesis project is performed in a team of two students whose names and contact details are listed in Table 3.1. In order to efficiently organize and manage the project, the Agile Management methodology will be utilized [31]. Agile is considered suitable for the current project since it involves regular iterations (sprints) which will give the opportunity to both students to have full overview of the studied field as well as gradually provide all the required deliverables.

Table 3. 1: Team members and contact details

Name	Email
Feidias Moulianitakis	feimou-7@student.ltu.se
Konstantinos Asimakopoulos	konasi-7@student.ltu.se

3.1 Lab Topology

Luleå University of Technology has provided the necessary equipment in order to install the required software and perform the evaluation tests of the developed benchmarking framework. The equipment consists of two virtual machines, each one with the specifications shown in Table 3.2.

Table 3. 2: Virtual machines specifications

Component	Capacity
Hard Disk Drive Capacity	300 GB
RAM	16 GB
CPU	Intel Xeon CPU E5-2690, 3GHz
Hosting Platform	ESXi to version 6.5

In both the virtual machines, Virtual Box 6.0.4 has been installed which hosts the required servers for the lab environment. The operating system of the two virtual servers is Windows Server 2019 Standard. In the first server, Microsoft SQL Server 2017 is installed [32] whereas the second server hosts Oracle MySQL 8.0.

SQL schema is the same in all machines. The databases were named according to their use namely “SQL_Unencrypted”, “SQL_Encrypted”, “MySQL_Unencrypted” and “MySQL_Encrypted”. Each database contains two tables, the “randominput10m” and “randominput1m”. The first, “randominput10m”, is used as the main table and is populated with 10 million records. The latter, “randominput1m”, is used as a table for INSERT query and is populated with 1 million records. The tables are created by a .csv file and a sample of this file is shown in Appendix I. The columns in each table are as follows:

Table 3. 3: Columns and Data types of the fields of the tables.

Column Name	Data Type
ID	int
FirstName	varchar
LastName	varchar
Age	int
CreditCardNo	bigint
Address	varchar
CityOfBirth	varchar
Gender	varchar

In Table 1 of Appendix I, the basic commands used for MS SQL are presented.
In Table 2 of Appendix I, the commands to enable TDE are presented for MS SQL.
In Table 3 of Appendix I, the basic commands used for MySQL are presented.
In Table 4 of Appendix I, the commands to enable TDE are presented for MySQL.

4. Benchmarking Framework

The goal of the benchmarking framework is to provide a methodology for measuring the performance impact of TDE to the processes of a database. The framework can be utilized for comparing different TDE systems from the performance perspective. Note that the security strength aspect of the TDE systems is out of scope for the current Thesis project. In order to achieve that, some core attributes of a database with the TDE feature enabled need to be identified and analysed.

To begin with, in order to interact with a database there is a number of queries. The most basic queries perform the core functions of a database which are the commands: SELECT, INSERT and DELETE. SELECT is used to find an entry in the database. INSERT is used to add a new entry in the database. DELETE is utilized, as its name indicates, to erase an entry from the database. By utilizing and combining the above queries, new more complicated queries can be derived performing specialized tasks. However, the impact of the encryption introduced by the TDE technology is clearly visible from those basic for the operation of a database queries.

A database has a number of performance metrics that indicate its efficiency and health state. The basic metrics are the Elapsed time, CPU time and hard disk memory consumption. The integration of a TDE system to a database is expected to have an impact on the aforementioned metrics. The encryption-decryption processes frequently require computationally intensive mathematical calculations which impose a burden to the system and in some cases might become a risk to the “transparency” feature of a TDE system.

Additionally to the above basic metrics, there is one more derived metric namely Throughput. The Throughput metric highlights the efficiency of the database and thus the TDE impact to its performance. The calculation of Throughput can be derived from the Elapsed time and the size of the table processed as shown in Section 2.2.

Furthermore, TDE systems have a number of user configurable parameters, two of which are expected to affect the performance of the system. These parameters are the encryption algorithm and the size of the symmetric keys. Already, in the related work section there has been a discussion comparing widely used symmetric encryption algorithms indicating that there are significant differences in the actual performance of encryption algorithms and increasing key-lengths [12].

Combining the above discussion, the following evaluation scenarios have been developed:

1. Evaluate the performance of TDEs regarding the Elapsed time (in seconds) for different: key lengths, encryption algorithms and table sizes.

2. Evaluate the performance of TDEs regarding the Throughput (in MB per second) for different: key lengths, encryption algorithms and table sizes.

3. Evaluate the performance of TDEs regarding the CPU time (processing time in seconds) for different: key lengths, encryption algorithms and table sizes.

4. Evaluate the performance of TDEs regarding the HD memory consumption (in MB) for different: key lengths, encryption algorithms and table sizes.

For the scope of the thesis project, AES and 3DES is used for encryption algorithms. The framework can be expanded with other encryption algorithms that are supported by the numerous TDE systems that are available on the market. Regarding the key length, for AES algorithm, the keys spaces are 128, 192 and 256 bits whereas for 3DES 168 bits, as already discussed in section 2.3. Finally, the table sizes span from a few thousand entries to a few millions.

The benchmarking framework consists of five distinct steps. The first step is the development of the topology. The lab topology is important since it needs to simulate the environment that needs the TDE system. For example, if we want to search for the best TDE system to protect a database with the accounts of a bank then a similar topology to the bank servers would be the most appropriate. However, for the scope of the current Thesis project, the lab topology is limited to two identical Virtual Machines (VMs). Each of these VMs is allocated to the two TDE systems that we want to benchmark and compare. In order to do that and have full control of the operating system, within each VM, two tier-2 VMs are created; one for benchmarking the database without the TDE feature enabled and the other with the TDE feature enabled.

After the creation of the lab topology, the second step is to build the databases and define the performance scenarios. Ideally, the databases should be copies of the real case scenario that needs to be protected by the TDE system. However, a simple database but with a significant number of entries would impose enough burden for stressing the TDE systems. Relatively, small databases without many entries will not provide visible results, especially for the metrics related to time and the values acquired would be biased by system interruptions and other processes that run in parallel. Furthermore, the designed scenarios should also replicate daily operations of the databases. For example, if a specific query is executed frequently, it should be included in the benchmarking scenarios.

The third step involves the activation of TDE. Currently, there is a plethora of TDE systems available on the market. Each of the TDE solutions has its own procedure for activating the TDE service. Hence, this step is case specific. In abstract terms, there are some common actions. The first is to generate the Database Master Key which is utilised to create a certificate in the master Database. Subsequently, this certificate is used to encrypt the Database Encryption Key which in turn, is used to encrypt the data stored in the user's database. Having finished with this process, TDE is ready for activation for the specific user database.

The fourth step of the framework is to execute the benchmarking scenarios, as previously discussed. In order to perform the benchmarking tests of these scenarios it is highly advisable

to use queries that simulate the daily operations run by the database users. However, benchmarking can be achieved with simple queries as well. The three basic queries that are suggested in order to perform benchmarking tests are SELECT, INSERT and DELETE. In order to improve the accuracy of the results each query is advisable to be executed for a significant number of times and then produce the average of the results. The automation of the process helps increase the number of executions and thus the accuracy of the final result.

Finally, the fifth and last step of the framework, the analysis of the results derived from the execution of the benchmarking scenarios. The results can be processed and presented in a series of graphs as it is performed in the current thesis project or in a raw from of table if this is convenient for the scope of each evaluation. However, the purpose is the same, the comparison of TDE systems from a performance perspective.

The above discussion and the five steps of the benchmarking framework are summarized in the following table:

Table 4. 1: Benchmarking Framework Steps

Steps	**Framework Procedure**
1	Creation of Topology - Lab Environment
2	Creation of Databases – Definition of Scenarios
3	Activation of TDE
4	Execution of Scenarios
5	Analysis of Results

5. Benchmarking Analysis

In the current Chapter, the results of the evaluation scenarios discussed in Chapter 4 are presented and analysed. The benchmarking framework is applied on two database suites: Microsoft SQL Server 2019 [32] and Oracle MySQL [33]. For the analysis, multiple executions of the most basic but yet frequently used queries were utilized namely, SELECT, DELETE and INSERT.

5.1 Scenario 1: TDE impact to the Elapsed time

As already discussed in Section 2.2, the Elapsed time is the actual time experienced by the user of a system. Its calculation starts when the user enters the command or the query and finishes when the result has been calculated and presented on screen. While the command or query is executed, other system processes might run in parallel affecting the recorded time. However, it is still an indication of the user experience. The impact of TDE to the Elapsed time can provide a metric of the transparency of such systems.

5.1.1 Key Length

One of the basic attributes of an encryption algorithm is the key length or key size. Almost all the modern encryption algorithms have configurable key lengths. Instant example, AES supports 128, 192 and 256 bit keys [34]. TDE systems, as already discussed, include the key length as a configuration parameter. For the scope of the current scenario AES algorithm is used with its three versions of keys (i.e. 128, 192 and 256 bit) since it is an algorithm commonly supported by the two TDE systems under comparison.

The following series of histograms indicate the impact of TDE to the Elapsed time for the three basic queries of a database, SELECT, DELETE and INSERT. The queries were executed for 1 million entries upon a table of 10 million entries. Note that the absolute numbers of these histograms have been included in Appendix II.

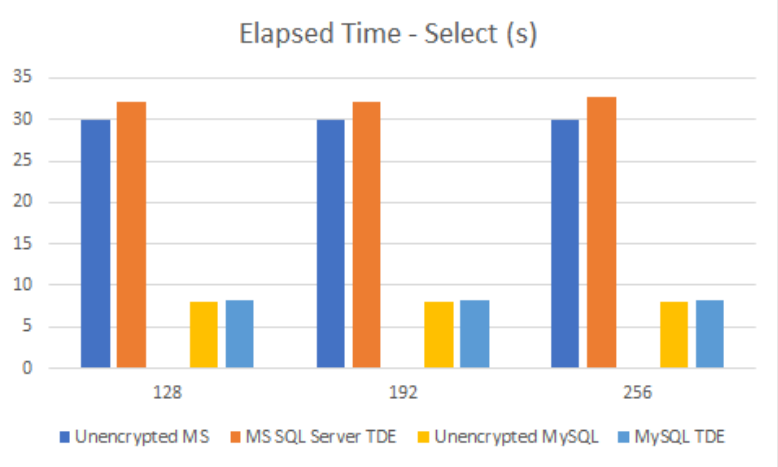

Figure 5. 1: Elapsed time for different key lengths (Select)

Observing the three histograms, it can be clearly stated that there is an impact of TDE to the total performance of a database. The SELECT query, consumes about 7% longer time for Microsoft SQL Server 2019 whereas for MySQL it is only half, about 3.5%. The DELETE query is affected more severely with MS SQL Server to last almost 4 seconds longer when TDE is enabled

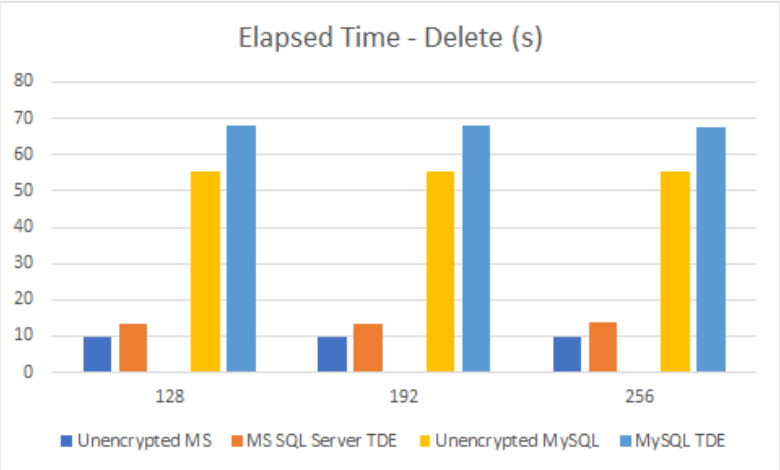

Figure 5. 2: Elapsed time for different key lengths (Delete)

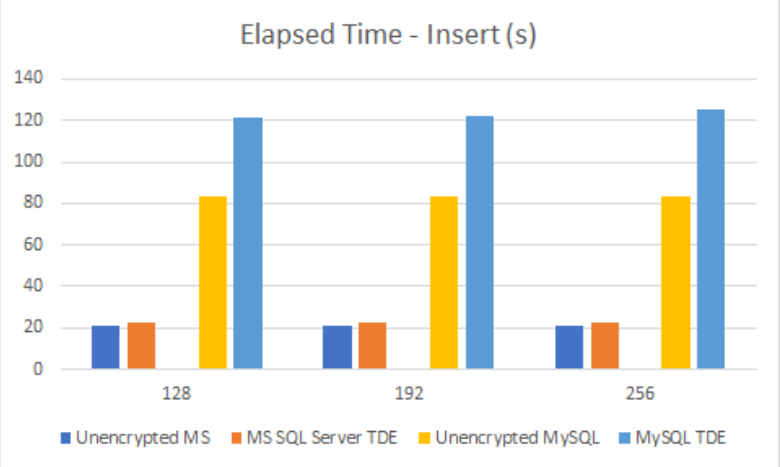

Figure 5. 3: Elapsed time for different key lengths (Insert)

whereas MySQL increases its Elapsed time by 12 seconds. Similarly, it can be observed that MS SQL Server 2019 performs better with the INSERT query which is affected by 5.4% and MySQL to be affected by about 46.6%.

Another point that can be derived from the three figures is that the implementation of each database affects significantly the performance. It is clearly visible that for the SELECT query, MySQL requires one fourth of the time needed by MS SQL Server. On the other hand, the latter, has significantly better performance from its opponent when it comes to DELETE and INSERT queries lasting 5 and 5.5 times less than the total time required by MySQL.

For this scenario, the TDE system that has the lowest impact to execution of the queries has the best performance. Hence, MS SQL Server 2019 has better Elapsed Time performance.

To sum up, with the current subscenario it has been highlighted that there is an impact to the Elapsed time with the introduction of TDE. The increase of the key length provides some fluctuation on the results but it does not create an increasing or decreasing trend. Finally, Elapsed time is subject to the architecture of each vendor as it has been observed by the differences between the two examined TDE systems for the DELETE and INSERT queries.

5.1.2 Encryption Algorithm

Another important attribute of a TDE system is the encryption algorithm it utilizes. Most of the TDE systems available on the market support a number of encryption algorithms. For example MS SQL Server 2019 supports AES and 3DES [35]. However, MySQL does only support AES encryption algorithm and thus will be excluded from this scenario [36].

The following series of pie charts indicate the impact of TDE to the Elapsed time for the three basic queries of a database, SELECT, DELETE and INSERT using different encryption algorithms (AES-128 and 3DES). The queries were executed for 1 million entries upon a table

of 10 million entries. Note that the absolute numbers of these pie charts have been included in Appendix II.

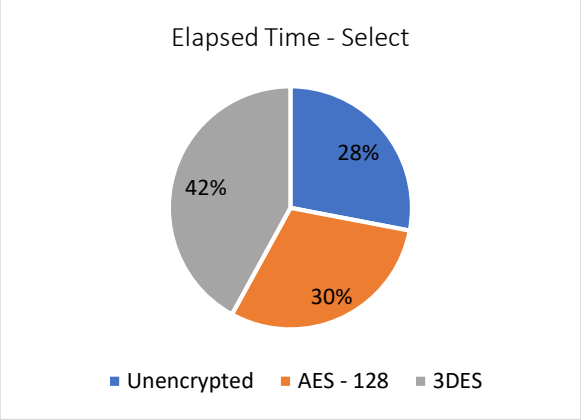

Figure 5. 4: Elapsed time for different encryption algorithms (Select)

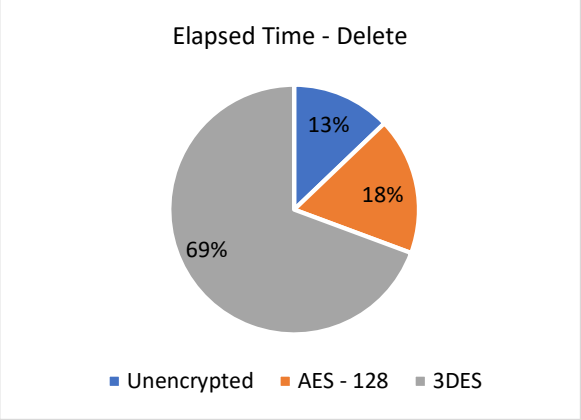

Figure 5. 5: Elapsed time for different encryption algorithms (Delete)

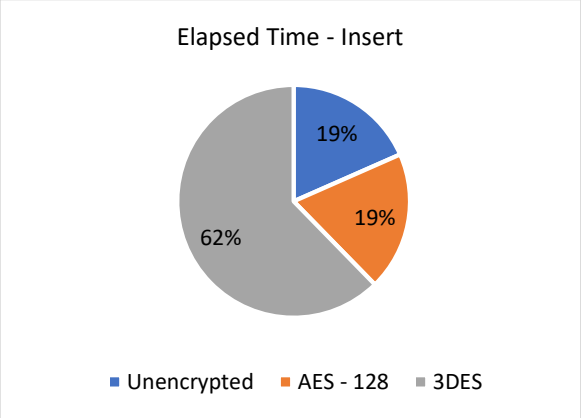

Figure 5. 6: Elapsed time for different encryption algorithms (Insert)

The three figures highlight the different impact of the encryption algorithm utilized by the TDE processes to the Elapsed time. It is clearly visible that 3DES is more time-consuming algorithm than AES. Specifically, 3DES consumes 40% more time than AES for the SELECT query as well as 3.83 and 3.26 times more for the DELETE and INSERT queries respectively. Another point that is observed from the figures is that AES performance is very close to the performance of the database without TDE.

Both the two highlighted points can be explained by the fact that AES is a widely used encryption algorithm that has been hardcoded to most of the modern CPUs [37]. Hence, its performance is closer to the unencrypted database compared to the time consumed by 3DES.

5.1.3 Table size

Databases grow in time as new entries are inserted. Protecting a database with TDE should take into account the scalability. Hence, with this sub-scenario, the TDE system scalability is benchmarked from a performance point of view. This is achieved by executing the three basic queries, SELECT, DELETE and INSERT for an increasing amount of entries for the two TDE systems under examination using the AES-128 encryption algorithm.

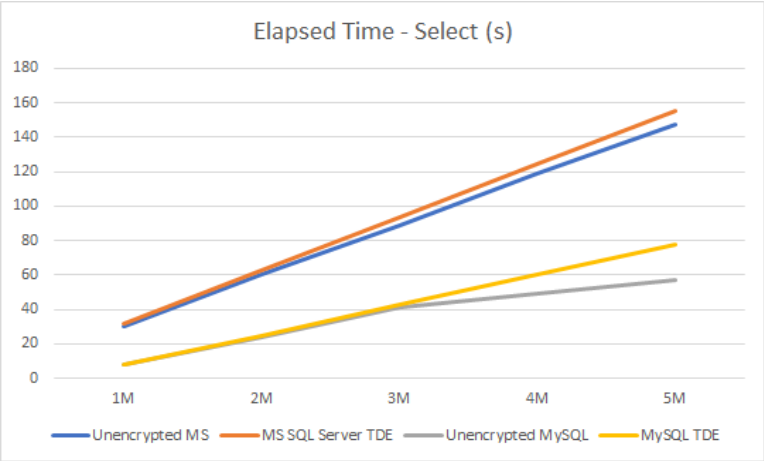

Figure 5. 7: Elapsed time for different Table Sizes (Select)

As it can be observed, the Elapsed time increases linearly with the increasing number of entries for both encrypted and unencrypted databases. Additionally to the above, the "transparency" of the operation is also visible in these series of figures since the encrypted version of the database with TDE slightly overtakes in the Elapsed time its respective unencrypted version.

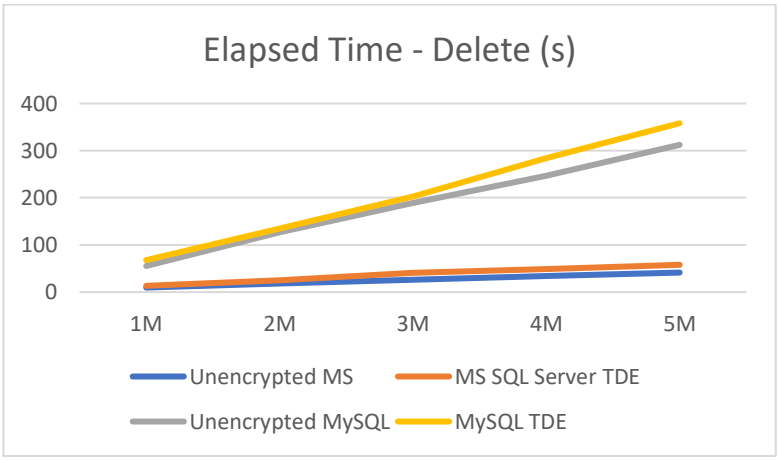

Figure 5. 8: Elapsed time for different Table Sizes (Delete)

Comparing the two database suites, it is clear that MS SQL Server 2019 is 3.75 times faster than MySQL for the SELECT query which is the only query in the experiment that has print results. The other two queries, DELETE and INSERT, do not have an output and there it is evident that MS SQL Server 2019 is 6.5 and 7.5 times faster than MySQL respectively.

Finally, it should be stated that when the number of entries increases and thus the Elapsed time increases, the interference from background processes running in parallel affect the query execution. Hence, for the accuracy of this experiment, it is required to run the test multiple times in order to absorb the deviation.

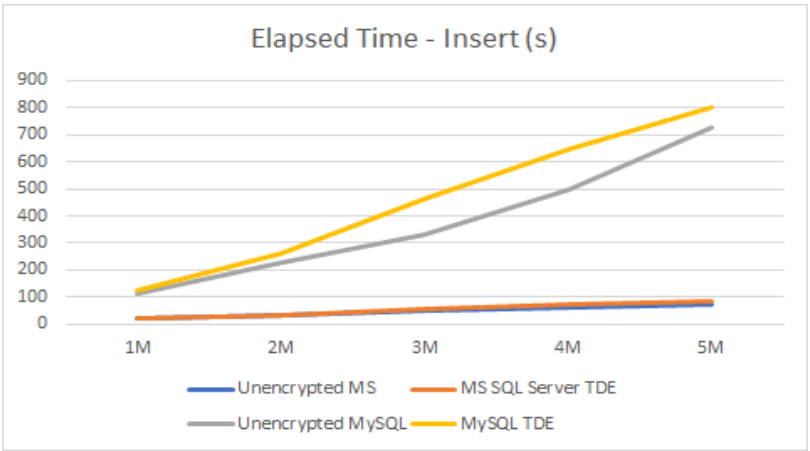

Figure 5. 9: Elapsed time for different Table Sizes (Insert)

5.2 Scenario 2: TDE impact to the Throughput

Elapsed time can be utilized to calculate the Throughput of the TDE operations. The formula for the calculation of Throughput has been discussed in Section 2.2 (formula 1). The Throughput metric highlights the efficiency of the database and thus the TDE impact to its performance. For the scope of the current scenario only the INSERT query is evaluated for different key lengths, encryption algorithms and increasing table size.

The following figure presents the achieved Throughput of the two examined databases with the TDE feature enabled and disabled. It is again noticeable that the TDE does not degrade significantly the achieved throughput of the database where it is applied. More specifically, for MS SQL Served 2019 the TDE impact is 0.23 Mbps whereas for MySQL is about 0.35 Mbps. However, a significant difference that is highlighted in this figure as well is between the two platforms. MS SQL Server 2019 is almost 4 and 5.5 times faster than MySQL in terms of Throughput with the TDE feature disabled and enabled respectively.

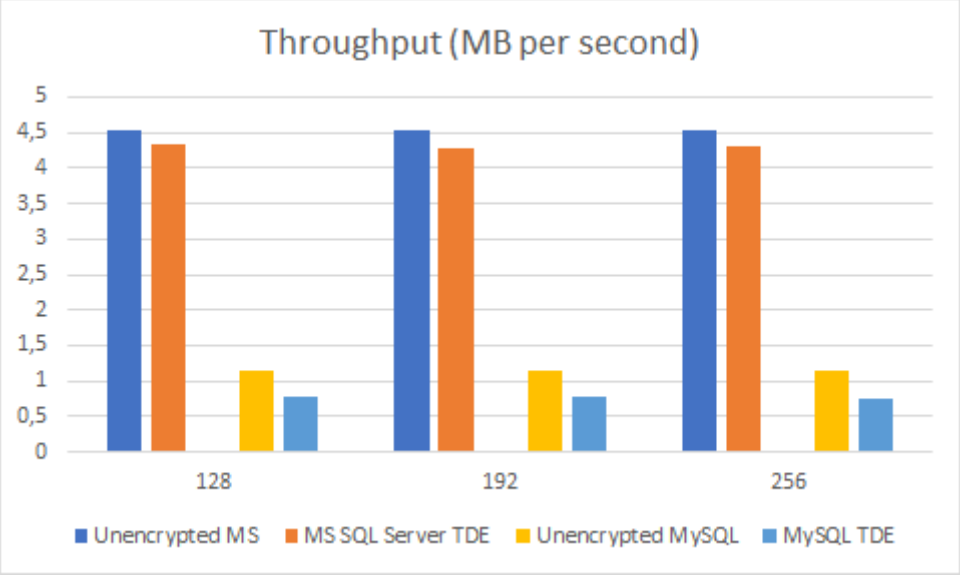

Figure 5. 10: Throughput for different Key Lengths

Subsequently, the following figure compares the Throughput provided by utilizing two different encryption algorithms for the purpose of TDE. The first is AES-128 and the latter is 3DES. It is clear that AES performance is very close to the performance of the unencrypted database. This, as explained before, is the result of the fact that AES is accelerated by the hardware. Furthermore, it can be observed that AES is 3.2 times faster than 3DES following the results of 5.1.2.

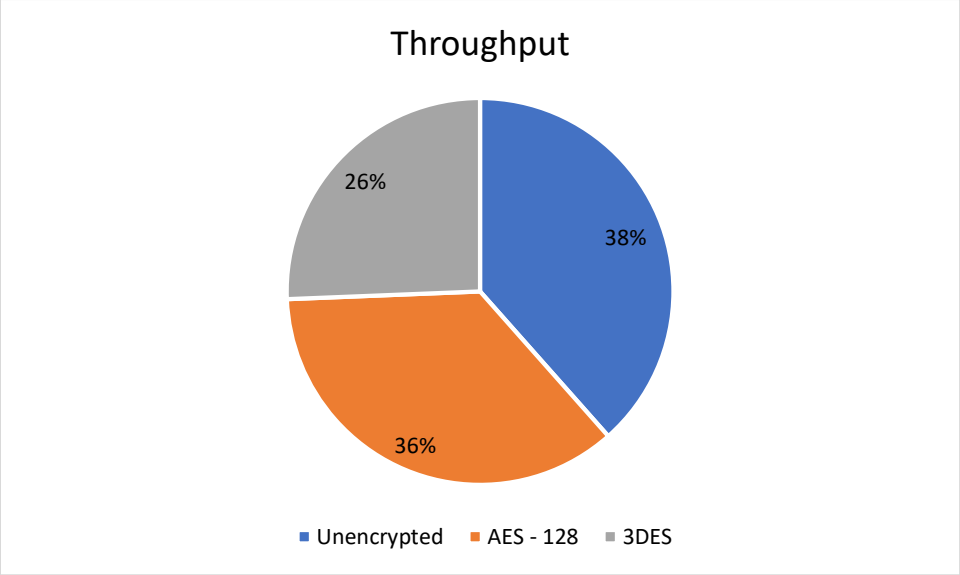

Figure 5. 11: Throughput comparison between AES-128 and 3DES

Finally, the next figure presents the impact of the increasing number of entries included in the INSERT query to the Throughput achieved by the TDE system. Again, the Unencrypted version of the database has slight better performance from the database with the TDE feature enabled. However, there is still a significant difference between the two systems under comparison. MS SQL server has better performance which starts with 5.6 times higher throughput than MySQL. This difference decreases as the number of entries increases and reaches the difference of about 1MB per second for 5 million entries.

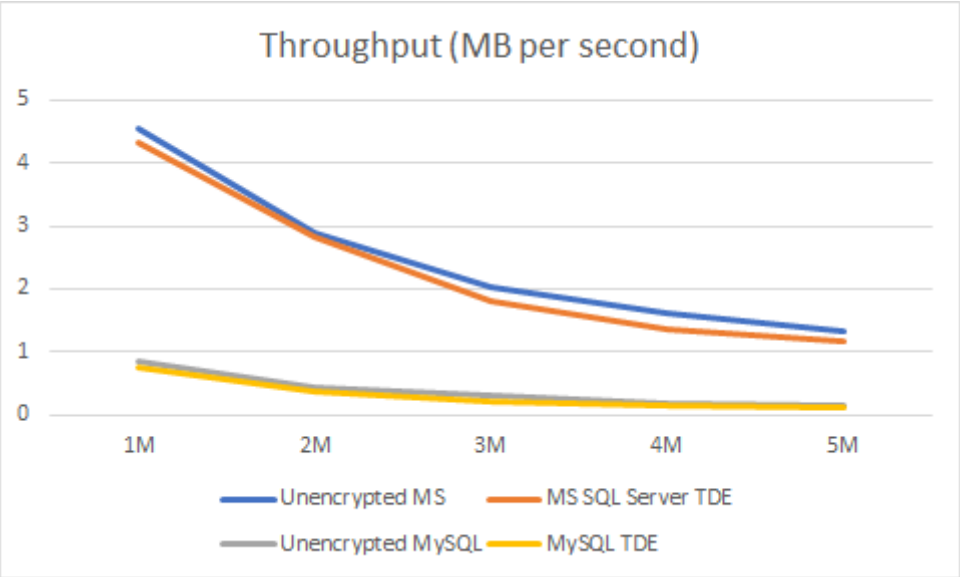

Figure 5. 12: Throughput comparison when Table Size increases

5.3 Scenario 3: TDE impact to the CPU time

The Elapsed time is a metric that indicates the user experience of a TDE system. However, it can be biased by other processes running in the background. Hence, the metric that actually reveals the burden introduced by the TDE to the system is the CPU time. CPU time, as discussed in Section 2.2, is defined as the time that the CPU spends on executing the commands of a specific process. As a result, in the current scenario, the impact of TDE to the CPU time is examined.

5.3.1 Key Length

In the current sub-scenario, the impact of the increase in the encryption key length to the CPU time is evaluated. As already discussed in 5.1.1, the AES algorithm is used with the three supported key length versions (i.e. 128, 192 and 256 bit).

The following series of histograms indicate the impact of TDE to the CPU time for the three basic queries of a database, SELECT, DELETE and INSERT. The queries were executed for 1 million entries upon a table of 10 million entries. Note that the absolute numbers of these histograms have been included in Appendix II.

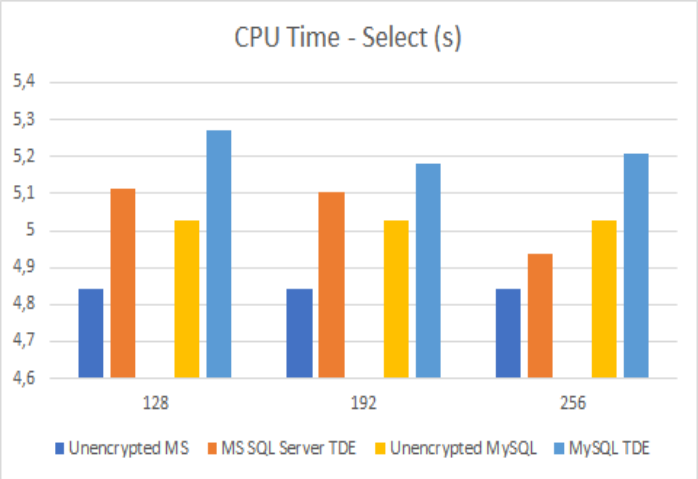

Figure 5. 13: CPU time for different key lengths (Select)

It can be derived from the histograms that there is an increase in the CPU time when TDE is enabled. The additional burden, in the SELECT query is about 4%, both in MS SQL Server and MySQL. Taking into account the absolute numbers, it can be observed that the additional CPU time consumed is less than a second. Hence, the impact of TDE can be considered negligible. When it comes to the DELETE query the results are similar for the MS SQL Server where the impact remains less than a second but it is observed that for MySQL there is a more significant impact which reaches 16.5% increase in the consumed CPU time. Moving to the third histogram, a similar pattern can be observed. MS SQL Server's impact remains less than a second whereas for MySQL the additional burden of 38 seconds.

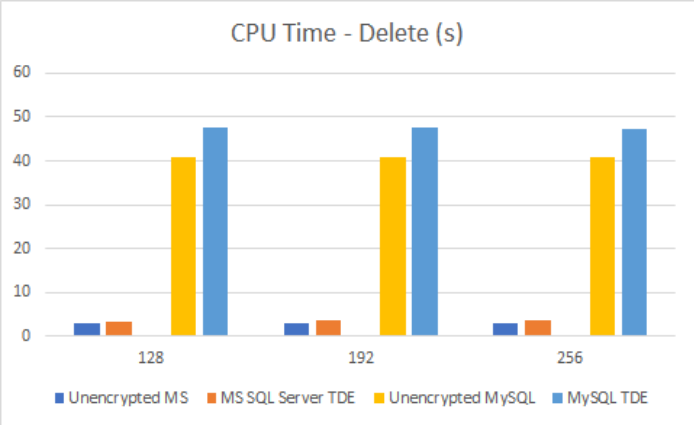

Figure 5. 14: CPU time for different key lengths (Delete)

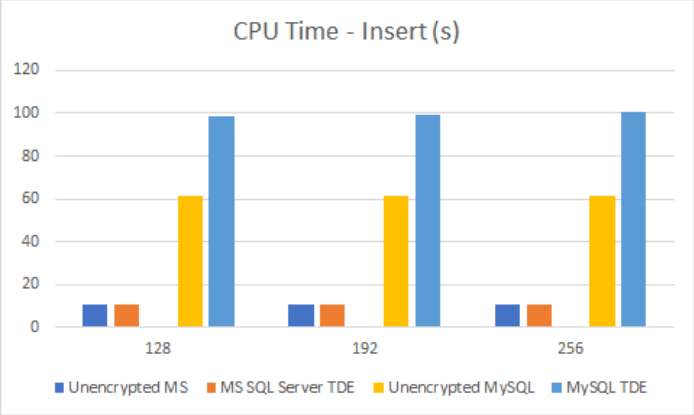

Figure 5. 15: CPU time for different key lengths (Insert)

Additionally, another point of comparison is the CPU time performance between the two TDE systems. For the SELECT query their differencies are negligible since they both require around 5 seconds to complete it. However, DELETE and INSERT cases are quite different. It can be clearly observed that MySQL requires significantly more CPU resources than MS SQL Server 2019. Hence, for the examined subscenario, it can be stated that MS SQL Server 2019 is more efficient than MySQL.

To sum up, with the current subscenario it has been highlighted that there is an impact to the CPU time with the introduction of TDE. The increase of the key length provides some fluctuation on the results but it does not create an increasing or decreasing trend.

Finally, CPU time is subject ot the architecture of each vendor as it has been observed by the differences between the two examined TDE systems for the DELETE and INSERT queries.

5.3.2 Encryption Algorithm

TDE can be configured with different encryption algorithms. Most of the TDE systems that are available on the market provide a variety of encryption algorithms to choose from. For example, MS SQL Server 2019 supports AES and 3DES [33]. However, MySQL does only support AES encryption algorithm and thus will be excluded from this scenario [34].

The following series of pie charts indicate the impact of TDE to the CPU time for the three basic queries of a database, SELECT, DELETE and INSERT using different encryption algorithms (AES-128 and 3DES). The queries were executed for 1 million entries upon a table of 10 million entries. Note that the absolute numbers of these pie charts have been included in Appendix II.

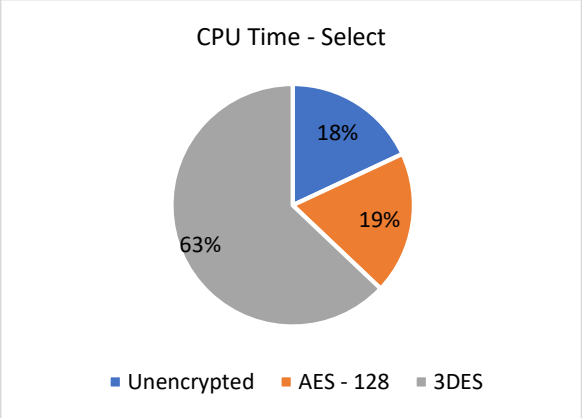

Figure 5. 16: CPU time for different encryption algorithms (Select)

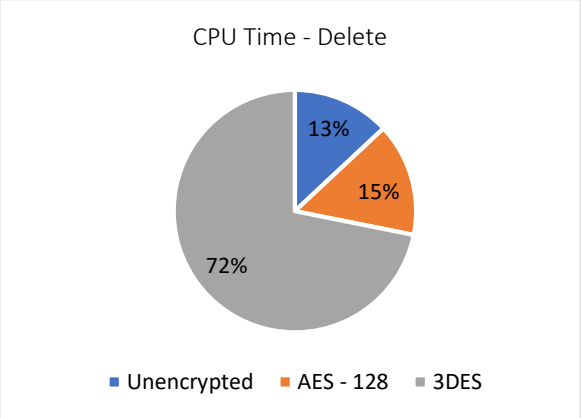

Figure 5. 17: CPU time for different encryption algorithms (Delete)

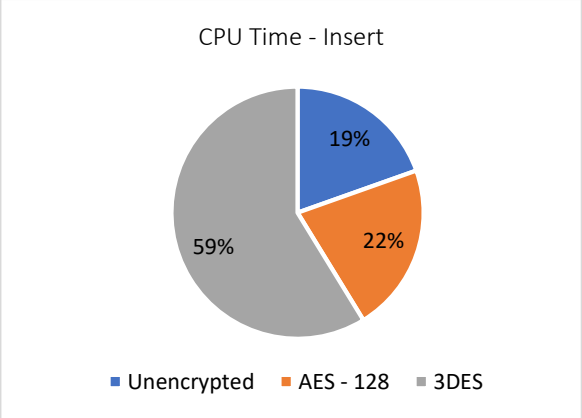

Figure 5. 18: CPU time for different encryption algorithms (Insert)

As can be observed from the figures, in this subscenario, like the sub-scenario of section 5.1.2, the 3DES algorithm is taking longer time than AES to complete. Depending on the query executed, 3DES consumes 2.7 to 4.8 times more CPU time than AES. Additionally, CPU time for AES is relatively similar to the unencrypted version of the database. Both these observations can be explained by the fact that AES is a widely used encryption algorithm that has been hardcoded to most of the modern CPUs [37].

5.3.3 Table size

Databases grow in time as new entries are inserted. Protecting a database with TDE should take into account the scalability. Hence, with this sub-scenario, the TDE system scalability is benchmarked from a performance point of view. This is achieved by executing the three basic queries, SELECT, DELETE and INSERT for an increasing amount of entries for the two TDE systems under examination using the AES-128 encryption algorithm.

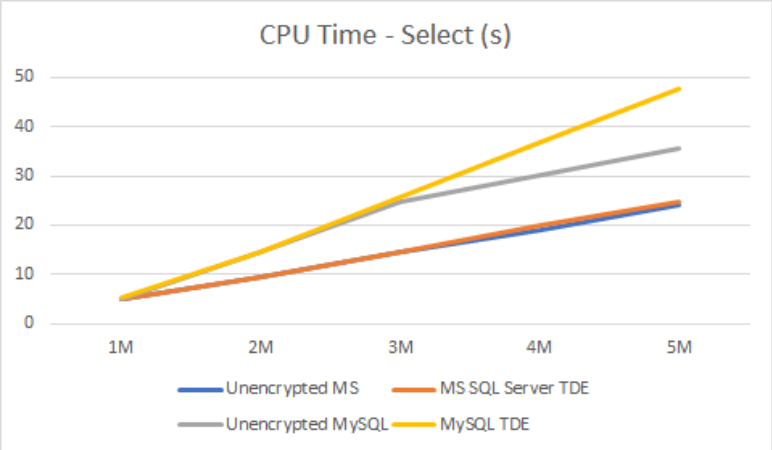

Figure 5. 19: CPU time for different Table Size (Select)

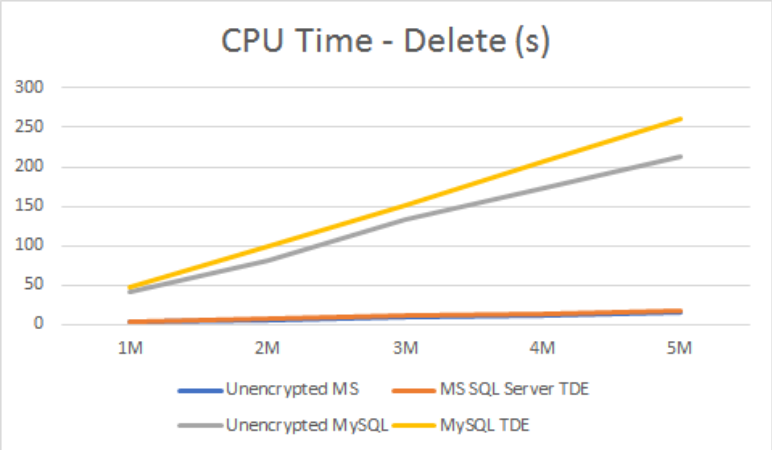

Figure 5. 20: CPU time for different Table Size (Delete)

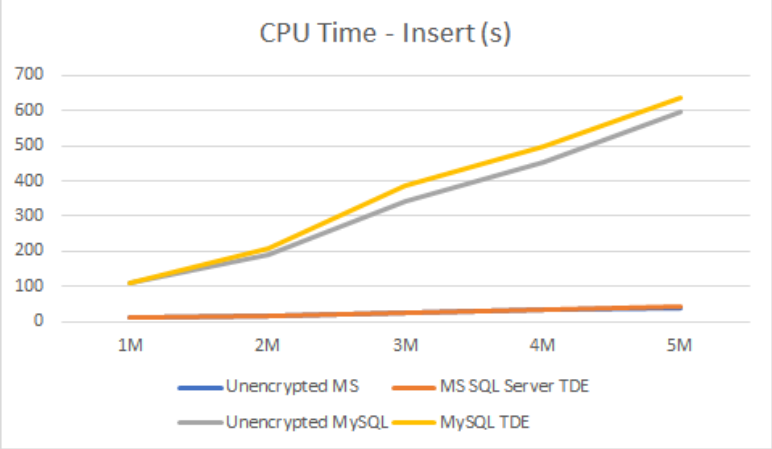

Figure 5. 21: CPU time for different Table Size (Insert)

As it can be observed, the Elapsed time increases linearly with the increasing number of entries for both encrypted and unencrypted databases. Additionally to the above, the "transparency" of the operation is also visible in these series of figures since the encrypted version of the database with the TDE feature slightly overtakes in the CPU time its respective unencrypted version.

Comparing the two database suites, it is observed that MySQL consume significantly more CPU time than MS SQL Server 2019. For the SELECT query MySQL performs 1.92 times slower than MS SQL Server 2019 when requesting 5 million entries. However, this difference increases even more when it comes to the DELETE and INSERT queries where MySQL is 15.26 and 15.50 times more CPU resource consuming than MS SQL Server 2019 respectively.

5.4 Scenario 4: TDE impact to the HardDisk Space

Databases are structures stored in HardDisk. Hence, applying TDE to a database could potentially affect the HardDisk space allocated to the database. However, multiple executions of the three basic queries have not resulted in any modification of the reserved HardDisk space. Hence, in this way the claim of many TDE vendors that their systems do not increase the size of the encrypted databases is verified [15] [38].

The actual data from the query executions are briefly presented in the following table. Note that in both databases, the same text table was imported having a size of 740MB. This was translated in a database of 964MB in Microsoft SQL Server 2019 and 955MB in MySQL. Despite the multiple encryption-decryption processes either for the execution of the queries or the activation-deactivation of the TDE, the total table size of the database remained stable.

Table 5. 1: Total table size in databases (in MB)

	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
Key length	964	964	955	955
Algorithm	964	964	955	955
Query size	964	964	955	955

6. Discussion

The presented results in Chapter 5 have revealed some interesting facts. To begin with, TDE introduces computational burden to the system since it applies encryption to the operations of the database. However, the impact of the encryption is considered of minor importance and especially for the CPU time metric where TDE performance was always close to the performance of the unencrypted version of the database. Hence, this fact validates the theory and the claim of the TDE vendors as discussed in Section 1.2 that the TDE feature is not only "transparent" due to the automatic key management from the system but also from the performance perspective.

Another interesting point that can be raised from the results is the significant difference in the performance for different encryption algorithms. AES encryption algorithm achieves significantly better performance from its opponent 3DES for the scenarios investigated in the scope of the present project. As already mentioned, this can be explained by the fact that AES is hardware accelerated by modern CPUs. Hence, this point validates the results of the papers reviewed in the literature study which illustrate the improved performance of AES over 3DES [12][14].

Last but not least, an interesting point that is also derived from the results presented in the previous Chapter is that the excecussion of the scenarios follow similar patterns for different TDE systems. Of course, there is some variation due to the fact that each vendor follows its own implementation and architecture but the patterns are still the same. This fact along with the other points discussed previously is a proof of validity for the processes of the developed theoretical benchmarking framework. Hence, the second goal of the present Thesis Project, the validation of the framework by applying it on real TDE systems, has been fullfiled.

Finally, as already discussed, there are multiple vendors with their own solutions and architectures. The performance of each product might vary. However, the general pattern of the impact of TDE feature should not be affected significantly since the feature's core operation is the encryption/decryption processes which are based on common algorithms. A noticeable impact to our discussion is expected to be imposed by differencies in the topology where different servers are introduced for distributing the encryption and key management functionalities. This distribution is expected to introduce extra time delays due to the network involved for connecting the servers. However, this is left as future work in the following subsection 6.2.

6.1 Answer to the Research Question

The Research Question the present Master Thesis Project investigates along with the answer provided by the study is as follows:

➢ How to methodically evaluate the performance of Transparent Data Encryption systems?

The present study deals with the adressed Research Question by developing a theoretical benchmarking framework. The framework provides a methodic five step procedure that can be utilized as a step by step guide. Following the steps, a user of the framework who can be academic personnel or a representative of a company that wants to invest in such systems, can create the lab environment (topology and databases), define the benchmarking scenarios, enable the TDE feature, execute the performance evaluation tests and finaly analyse the results. The framework is flexible and can be adjusted to the needs of different use cases in order to meet the demands of different enterprises. Finally, the validity of the framework has been proven by applying it on real TDE systems for simple but computentional intensive scenarios.

6.2 Future Work

The master thesis project has attempted to cover a variety of scenarios, however, due to the limited timeframe and the available resources, there are some improvements that could be introduced to the developed benchmarking framework that are left as future work.

To begin with, the current framework focuses only on the performance aspect of the TDE systems. However, the security strength of the system could also be a part of the evaluation of such systems. Hence, the developed framework could be extended with penetration testing.

The evaluation scenarios were conducted manually, multiple times and then the results were averaged in order to improve the accuracy. Due to the fact that the process was conducted manually it was not feasible to conduct vast amount of executions. Hence, it is suggested as future work to automate the execution process. In this way, the executions can be significantly extended and thus obtain even more accurate results.

Finally, due to the limited resources the developed framework has been evaluated only on virtual machines. As a future work is suggested to create a lab testbed with physical machines in order to perform the evaluation of the framework. In this way, the framework could be validated, also for the scenarios where a company deploys the TDE system in physical machines.

7. Conclusion

Data play a significant role in business operations as well as our personal lives. Thus, it is of crucial importance to protect them. The generated data need to be stored and organized in order to be always available to their users. This is achieved by storing them in an organized manner in databases. As a result, it is important to protect them while at rest and according to the literature, TDE is the most efficient way. The main benefit of TDE is that it protects the data in a transparent way, which means that the user of the system does not have to take care of the encryption processes or the key management.

The purpose of the current thesis project is to methodically evaluate the performance of Transparent Data Encryption systems. This is achieved by developing a theoretical benchmarking framework that focuses on the evaluation of the performance of TDE systems. The framework procedure involves four steps which are the creation of the lab environment, the creation of the databases, the activation of the TDE feature and finally, the execution of the benchmarking scenarios. Multiple scenarios can be developed as part of this benchmarking framework depending on the current needs. However, for the scope of this thesis project the scenarios have been derived from the basic performance metric, which is time. Hence, time has been divided into two categories, Elapsed time and CPU time. Additionally, throughput has been evaluated as well as hard disk space.

The developed theoretical benchmarking framework has been evaluated by applying it for the comparison of two different databases that support TDE. The framework succeeds in contrasting the differences in the implementation of the two systems as well as highlight the additional computational burden introduced, when the TDE feature is enabled. As a result, the product of the present master thesis project can be utilised in order to compare TDE systems and define the configuration that better applies to each deployment situation.

7.1 Acknowledgements

We have kept this last section of our Thesis project to deliver our acknowledgements to Tieto AB Sweden. Tieto is an innovative firm with interests in academic research. One example of Tieto’s support was the current Master Thesis project. The project itself was an inspiration of Tieto. However, it did not just come up with a general idea. Tieto’s representative was always available to assist us with any issues that we faced as well as provided us with contact details of TDE system vendors. Hence, we would like to conclude this report by expressing our thanks to Tieto for their inspiration and support.

References

[1] Min Xu, Jeanne M. David, Suk Hi Kim, “The Fourth Industrial Revolution: Opportunities and Challenges.”

[2] Statista, “Share of households with a computer worldwide 2005-2018 | Statistic,” *Statista*. [Online]. Available: https://www.statista.com/statistics/748551/worldwide-households-with-computer/. [Accessed: 31-Mar-2019].

[3] Internet World Stats, “World Internet Users Statistics and 2019 World Population Stats.” [Online]. Available: https://www.internetworldstats.com/stats.htm. [Accessed: 01-Apr-2019].

[4] Elmasri, Ramez, Navathe, Shamkant, *Fundamentals of Database Systems, 5th Edition 5th*, 5th edition. Pearson / Addison Wesley.

[5] Study, “What is Transparent Data Encryption (TDE)?,” *Study.com*. [Online]. Available: http://study.com/academy/lesson/what-is-transparent-data-encryption-tde.html. [Accessed: 02-Mar-2019].

[6] Thales Security, “What is Transparent Encryption? | Thales eSecurity.” [Online]. Available: https://www.thalesesecurity.com/faq/encryption/what-transparent-encryption. [Accessed: 23-Jan-2019].

[7] Oracle, “Database Advanced Security Guide.” [Online]. Available: https://docs.oracle.com/database/121/ASOAG/introduction-to-transparent-data-encryption.htm#ASOAG10117. [Accessed: 18-Aug-2019].

[8] MySQL, “MySQL :: MySQL Enterprise Transparent Data Encryption (TDE).” [Online]. Available: https://www.mysql.com/products/enterprise/tde.html. [Accessed: 18-Aug-2019].

[9] S. Mukherjee, “Popular SQL Server Database Encryption Choices,” *arXiv:1901.03179 [cs]*, Jan. 2019.

[10] D. A. P. Deshmukh and D. R. Qureshi, “Transparent Data Encryption -- Solution for Security of Database Contents,” Mar. 2013.

[11] D. S. A. Elminaam, H. M. A. Kader, and M. M. Hadhoud, “Performance Evaluation of Symmetric Encryption Algorithms,” p. 7, 2008.

[12] T. Hardjono and L. R. Dondeti, *Security In Wireless LANS And MANS*. Boston: Artech House Print on Demand, 2005.

[13] Abdel-Karim Al-Tamimi, “(PDF) Performance Analysis of Data Encryption Algorithms,” *ResearchGate*. [Online]. Available: https://www.researchgate.net/publication/228775009_Performance_Analysis_of_Data_Encryption_Algorithms. [Accessed: 30-Nov-2018].

[14] J. Hu and A. Klein, “A Benchmark of Transparent Data Encryption for Migration of Web Applications in the Cloud,” in *2009 Eighth IEEE International Conference on Dependable, Autonomic and Secure Computing*, 2009, pp. 735–740.

[15] Aliceku, “Transparent Data Encryption (TDE) - SQL Server.” [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption. [Accessed: 17-Mar-2019].

[16] Don Kiely, “Meet the Data Protection API,” *IT Pro*, 30-Oct-2009. [Online]. Available: https://www.itprotoday.com/development-techniques-and-management/meet-data-protection-api. [Accessed: 03-May-2019].

[17] VanMSFT, “SQL Server Certificates and Asymmetric Keys - SQL Server.” [Online]. Available: https://docs.microsoft.com/en-us/sql/relational-databases/security/sql-server-certificates-and-asymmetric-keys. [Accessed: 03-May-2019].

[18] GNU, “Processor And CPU Time (The GNU C Library).” [Online]. Available: https://www.gnu.org/software/libc/manual/html_node/Processor-And-CPU-Time.html#Processor-And-CPU-Time. [Accessed: 03-May-2019].
[19] Archiveddocs, “Overview of Performance Monitoring.” [Online]. Available: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc958260(v%3dtechnet.10). [Accessed: 03-May-2019].
[20] Study, “What is Elapsed Time? - Definition & Examples - Video & Lesson Transcript,” *Study.com*. [Online]. Available: http://study.com/academy/lesson/what-is-elapsed-time-definition-examples.html. [Accessed: 03-May-2019].
[21] Cryptopp, “Benchmarks - Crypto++ Wiki.” [Online]. Available: https://www.cryptopp.com/wiki/Benchmarks#Benchmark_Metrics. [Accessed: 31-Jul-2019].
[22] J. Schwartz, “TECHNOLOGY; U.S. Selects a New Encryption Technique,” *The New York Times*, 03-Oct-2000.
[23] NIST, “FIPS 197, Advanced Encryption Standard (AES),” p. 51.
[24] J. Daemen, “The Rijndael Block Cipher,” p. 47.
[25] P. Karn, W. A. Simpson, and P. Metzger, “The ESP Triple DES Transform.” [Online]. Available: https://tools.ietf.org/html/rfc1851. [Accessed: 04-May-2019].
[26] Cryptosys, “Triple DES.” [Online]. Available: https://www.cryptosys.net/3des.html. [Accessed: 01-May-2019].
[27] K. Peffers, T. Tuunanen, M. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of Management Information Systems*, vol. 24, pp. 45–77, Jan. 2007.
[28] A. R. Hevner, “Design Science in Information Systems Research 1,” 2004.
[29] P. Marshall and J. McKay, “The dual imperatives of action research,” *Info Technology & People*, vol. 14, no. 1, pp. 46–59, Mar. 2001.
[30] M. Sein, O. Henfridsson, S. Purao, M. Rossi, and R. Lindgren, “Action Design Research,” *Management Information Systems Quarterly*, vol. 35, no. 1, pp. 37–56, Mar. 2011.
[31] H. F. Cervone, “Understanding agile project management methods using Scrum,” *OCLC Systems & Services*, vol. 27, no. 1, pp. 18–22, Feb. 2011.
[32] Microsoft, “Server 2019 | Microsoft,” *Microsoft SQL Server - US (English)*. [Online]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-2019. [Accessed: 30-Mar-2019].
[33] MySQL, “MySQL.” [Online]. Available: https://www.mysql.com/. [Accessed: 16-Aug-2019].
[34] ENISA, “Algorithms, key size and parameters report 2014.” [Online]. Available: https://www.enisa.europa.eu/publications/algorithms-key-size-and-parameters-report-2014. [Accessed: 18-Aug-2019].
[35] Steve Stein, “Data Encryption in SQL Server.” [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/data/adonet/sql/data-encryption-in-sql-server. [Accessed: 18-Aug-2019].
[36] MySQL, “MySQL :: MySQL 8.0 Reference Manual :: 6.4.4.7 Supported Keyring Key Types.” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/keyring-key-types.html. [Accessed: 18-Aug-2019].
[37] K. P. Singh and S. Dod, “An Efficient Hardware design and Implementation of Advanced Encryption Standard (AES) Algorithm,” *International Journal of Recent Advances in Engineering & Technology*, vol. 4, no. 2, pp. 5–9, 2016.
[38] A. P. Deshmukh and R. Qureshi, “Transparent Data Encryption- Security of Database Using Microsoft SQL Server 2008 and Oracle,” 2011.

Appendix I

Commands used in MS SQL environment

Table I. 1: Commands used for MS SQL

	Command	**Instance**
1	Create	Create table Persons(PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255),);
2	Insert	SET STATISTICS TIME ON; INSERT INTO dbo.Final select * from dbo.RandomInput100k GO SET STATISTICS TIME OFF;
3	Select	SET STATISTICS TIME ON; select top 100000 * from Final; SET STATISTICS TIME OFF;
4	Delete	SET STATISTICS TIME ON; delete top 100000 * from Final; SET STATISTICS TIME OFF;
5	Table Size	sp_msforeachtable N'EXEC sp_spaceused [?]';

Table I. 2: Enable Transparent Data Encryption

	Command	**Instance**
1	Use Service Master Key to create the certificate for Database “Test1”	USE master; GO CREATE MASTER KEY ENCRYPTION BY PASSWORD = 'A7005E-LTU2019'; go CREATE CERTIFICATE MyServerCert WITH SUBJECT = 'My Certificate'; go USE Test1; GO

2	Choose Encryption Algorithm and the connected certificate	CREATE DATABASE ENCRYPTION KEY WITH ALGORITHM = AES_128 ENCRYPTION BY SERVER CERTIFICATE MyServerCert;
3	Enable Encryption in database “Test1”	ALTER DATABASE Test1 SET ENCRYPTION ON; GO

Commands used in MYSQL environment

Table I. 3: Commands used for MYSQL

	Command	**Instance**
1	Create	Create table Persons(PersonID int, LastName varchar(255), FirstName varchar(255), Address varchar(255), City varchar(255),);
2	Enable Statistics	Set profiling=1;
3	Insert	INSERT INTO mysql_tde.randominput10m SELECT * FROM mysql_tde.random_input_test LIMIT 4000000;
4	Select	SELECT SQL_NO_CACHE * FROM mysql_tde.randominput10m LIMIT 0, 1000000;
5	Delete	DELETE FROM mysql_tde.randominput10m LIMIT 4000000;
6	Table Size	SELECT table_name AS `Table`, round(((data_length + index_length) / 1024 / 1024), 2) `Size in MB` FROM information_schema.TABLES WHERE table_schema = "mysql_tde" AND table_name = "randominput10m";
7	Show the needed times	show profiles; SHOW PROFILE cpu FOR QUERY xx;

Table I. 4: Enable Transparent Data Encryption

	Command	**Instance**
1		SET GLOBAL default_table_encryption=ON;
	Enable TDE on tables and on schema	ALTER table randominput10m ENCRYPTION = 'Y'; ALTER table randominput1m ENCRYPTION = 'Y'; ALTER database mysql_tde encryption = 'Y';
2	Choose Encryption Algorithm	Server-> Status and System Variables->System Variables-> 'block_encryption_mode' = aes-128-cbc
3	Check if encryption is enabled	SELECT PLUGIN_NAME, PLUGIN_STATUS FROM INFORMATION_SCHEMA.PLUGINS WHERE PLUGIN_NAME LIKE 'keyring%'; SELECT SPACE, NAME, SPACE_TYPE, ENCRYPTION FROM INFORMATION_SCHEMA.INNODB_TABLESPACES WHERE ENCRYPTION='Y'; SELECT TABLE_SCHEMA, TABLE_NAME, CREATE_OPTIONS FROM INFORMATION_SCHEMA.TABLES WHERE CREATE_OPTIONS LIKE '%ENCRYPTION%';

Appendix II

Results

Scenario 1: TDE impact to the Elapsed time

SELECT

Table II. 1: SC1 – Select - Key Length

	Elapsed time - Select (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	30,004	32,140	7,999	8,239
192	30,004	32,179	7,948	8,223
256	30,004	32,702	7,948	8,227

Table II. 2: SC1 – Select - Encryption Algorithm

	Elapsed time - Select (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	30,004	32,140
3DES	30,004	45,010

Table II. 3:SC1 – Select - Table Size

	Elapsed time - Select (ms)			
Table Size	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	30,004	32,140	7,999	8,239
2M	60,6342	62,307	23,581	24,245
3M	88,6956	93,230	41,229	42,605
4M	118,7924	124,700	49,526	59,963
5M	147,2988	155,246	57,317	77,700

DELETE

Table II. 4: SC1 – Delete - Key Length

	Elapsed time - Delete (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	9,646	13,350	55,483	67,794
192	9,646	13,541	55,483	67,872
256	9,646	13,774	55,483	67,685

Table II. 5: SC1 – Delete - Encryption Algorithm

	Elapsed time - Delete (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	9,646	13,350
3DES	9,646	52,007

Table II. 6: SC1 – Delete - Table Size

	Elapsed time - Delete (ms)			
Table Size	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	9,646	13,350	55,483	67,794
2M	17,7808	24,604	126,470	134,779
3M	26,7018	40,641	188,629	202,974
4M	33,9056	48,842	247,347	284,593
5M	41,4408	57,902	312,293	358,046

INSERT

Table II. 7: SC1 – Insert - Key Length

	Elapsed time - Insert (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	21,250	22,272	83,815	121,178
192	21,250	22,533	83,815	121,817
256	21,250	22,404	83,815	125,599

Table II. 8: SC1 – Insert - Encryption Algorithm

	Elapsed time - Insert (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	21,250	22,272
3DES	23,650	71,947

Table II. 9: SC1 – Insert - Table Size

	Elapsed time - Insert (ms)			
Table Size	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	21,250	22,272	112,263	125,498
2M	33,4054	34,161	227,847	262,209
3M	47,4988	52,929	329,485	465,702
4M	60,0312	70,616	498,763	644,802
5M	71,9496	82,223	725,408	804,147

Scenario 2: TDE impact to the CPU time

SELECT

Table II. 10: SC2 – Select - Key Length

	CPU time - Select (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	4,844	5,112	5,025	5,272
192	4,844	5,106	5,025	5,087
256	4,844	4,938	5,025	5,209

Table II. 11: SC2 – Select - Encryption Algorithm

	CPU time - Select (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	4,844	5,112
3DES	4,844	16,897

Table II. 12: SC2 – Select - Table Size

	CPU time - Select (ms)			
Table Size	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	4,844	5,112	5,025	5,272
2M	9,4248	9,594	14,500	14,669
3M	14,6282	14,669	24,719	25,584
4M	19,0908	19,910	30,113	36,831
5M	24,0062	24,806	35,506	47,592

DELETE

Table II. 13: SC2 – Delete - Key Length

	CPU time - Delete (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	2,925	3,387	40,775	47,491
192	2,925	3,489	40,775	47,648
256	2,925	3,711	40,775	47,438

Table II. 14: SC2 – Delete - Encryption Algorithm

	CPU time - Delete (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	2,925	3,387
3DES	2,925	16,109

Table II. 15: SC2 – Delete - Table Size

	CPU time - Delete (ms)			
Table Size	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	2,925	3,387	40,775	47,491
2M	5,8312	6,916	81,844	98,367
3M	8,8812	10,875	132,250	151,611
4M	11,7718	13,878	172,023	207,736
5M	14,6094	17,109	212,818	261,233

INSERT

Table II. 16: SC2 – Insert - Key Length

	CPU time - Insert (ms)			
Key Length (bits)	Unencrypted MS SQL Server	MS SQL Server with TDE	Unencrypted MySQL	MySQL with TDE
128	10,215	10,570	61,628	98,956
192	10,215	10,778	61,628	99,285
256	10,215	10,675	61,628	100,889

Table II. 17: SC2 – Insert - Encryption Algorithm

	CPU time - Insert (ms)	
Algorithm	Unencrypted MS SQL Server	MS SQL Server with TDE
AES - 128	10,215	11,306
3DES	11,615	30,660

Table II. 18: SC2 – Insert - Table Size

	CPU time - Insert (ms)			
Key length	Unencrypted MS	MS SQL Server TDE	Unencrypted MySQL	MySQL TDE
1M	10,215	11,306	108,574	110,672
2M	16,3394	16,969	190,211	209,664
3M	23,422	23,963	343,161	386,663
4M	32,8594	34,482	454,271	496,494
5M	39,3092	41,000	598,297	635,637