Examensarbete

HÖGSKOLAN
I SKÖVDE

PERFORMANCE BENCHMARKING OF DATA-AT-REST ENCRYPTION IN RELATIONAL DATABASES

Bachelor Degree Project in Information Technology
Basic level 30 ECTS
Spring 2022

Stewart Istifan
Mattias Makovac

Supervisor: Yacine Atif
Examinator: Juhee Bae

Abstract

This thesis is based on measuring how Relational Database Management Systems utilizing data-at-rest encryption with varying AES key lengths impact the performance in terms of transaction throughput of operations through the process of a controlled experiment. By measuring the effect through a series of load tests followed by statistical analysis, the impact of adopting a specific data-at-rest encryption algorithm could be displayed. The results gathered from this experiment were measured regarding the average transactional throughput of SQL operations. An OLTP workload in the benchmarking tool HammerDB was used to generate a transactional workload. This, in turn, was used to perform load tests on SQL databases encrypted with different AES-key lengths. The data gathered from these tests then underwent statistical analysis to either keep or reject the stated hypotheses.

The statistical analysis performed on the different versions of the AES-algorithm showed no significant difference in terms of transaction throughput concerning the results gathered from the load tests on MariaDB. However, statistically, significant differences are proven to exist when running the same tests on MySQL. These results answered our research question, *"Is there a significant difference in transaction throughput between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL and MariaDB?"*. The conclusion is that the statistical evidence suggests a significant difference in transactional throughput between AES algorithms in MySQL but not in MariaDB.

This conclusion led us to investigate further transactional database performance between MySQL and MariaDB, where a specific type of transaction is measured to determine if there was a difference in performance between the databases themselves using the same encryption algorithm. The statistical evidence confirmed that MariaDB vastly outperformed MySQL in transactional throughput.

Keywords: Data-at-rest, Encryption, Performance comparison, Relational Database Management System, MariaDB, MySQL, Advanced Encryption Standard (AES).

Acknowledgements

We would like to thank our supervisor Yacine Atif at the University of Skövde for aiding us with valuable information and insight throughout our study. We would also like to thank Bally's Interactive in Skövde for providing us with the computer hardware and the office space needed to conduct our study.

Table of Contents

1 Introduction

Everything from search engines to email services utilizes a database storage engine to store and retrieve information. When sensitive data is stored, it is also crucial to adequately secure it. One way of securing data stored in a database, i.e., data that is at rest, is by encrypting it at the storage level. When doing so, the intent is to ensure that only authorized entities have access to the data. This problem can be solved by several methods, such as by using an encryption key to encrypt and decrypt the data.

There are several techniques for encrypting this form of stored data. The Advanced Encryption Standard (AES) block cipher is one of the most commonly used ciphers for symmetric key cryptography. The AES block cipher comes with a fixed block size of 128 bits and three different key lengths, 128-bit, 192-bit, and 256-bit, respectively. Deciding on which one of these key lengths to use for encrypting data and the performance impact it might have is not well established in current research in the context of relational database cryptography. If one is to decide whether to implement AES encryption on their stored data, knowing the trade-offs between implementing it with either of the three key lengths might come to question. By knowing which version of the AES algorithm provides the highest security level without compromising transaction performance, one can make a more informed decision. There are few studies published on this topic. However, none of them investigate or focus on the encryption algorithms used for data-at-rest implementation. Thus, this thesis focuses on experimenting with and comparing the performance differences of various key lengths of the popular AES algorithm used to encrypt data-at-rest.

This study aims to aid developers in implementing encryption in databases by providing theoretical data that can be helpful when evaluating the trade-offs in terms of performance when considering a specific AES encryption key length in relational databases.

2 Background

2.1 Key Concepts

In order to differentiate between similar abbreviations in this study, some key concepts will be briefly explained in the following subsections.

2.1.1 Database

In short, a database is simply an organized collection of data that is stored and accessed electronically. Retrieval or modification of data, also known as "queries" are sent to the database to access or modify data stored in the database.

2.1.2 Relational Database Management System

A database management system, also abbreviated as DBMS, is software that primarily functions as an interface between the end-users and the database itself, where the users of the system can perform several different kinds of operations for manipulating data or managing the overall structure of the database (IBM, no date).

A relational database management system, also abbreviated as RDBMS, is a type of database that stores data and creates access points between data related to each other. RDBMS is based on the relational model, which represents data in tables where each row in the table is an entry that can be identified with a unique ID, known as a key. Columns of the table store attributes of the data, and each entry has a value corresponding to each attribute. These factors make forming relationships between data points easier (Oracle, 2021).

2.1.3 Data-at-rest

Data-at-rest in the context of databases refers to data being flushed from memory and written to disk, contrary to data-in-transit, where the information is "in motion" and exchanged through a communication medium. Data-at-rest can be stored in several ways, i.e., in a database server, external backup mediums, computer, or mobile devices.

2.2 Transactions

In a database context, a transaction can be considered a sequence of operations sent to the database consisting of instructions that the database will execute. These transactions could be different tasks. Such a task could either modify or delete data that resides in the database. All Relational databases today assure data consistency, which means that the DBMS provides tools for ensuring data integrity (Meier and Kaufmann, 2019). In relational databases, a transaction ensures data integrity by adhering to principles known as ACID.

In their book, Meier and Kaufmann (2019) elaborate on the ACID principles and how they allow conflict-free simultaneous work between different users. A shortlist of these principles is explained below.

- **Atomicity** (A) - Transactions are either applied fully or not at all.

- **Consistency** (C) - Integrity constraints may be temporarily violated during the transaction. However, a transaction should always move the database from one consistent state into another upon completion.
- **Isolation** (I) – Isolates individual transactions from other transactions executed simultaneously, protecting them from unwanted side effects.
- **Durability** (D) – The state of a database must remain valid and be maintained until a transaction changes it in the case of a system failure or other software errors.

2.3 Database Encryption

Database encryption functions similarly to general encryption algorithms where data visible in plain text is converted into ciphertext. Tilborg and Jajodia (2011) define database encryption as techniques used to transform a plain text database into an encrypted database, making the content unreadable except for those who have access to the encryption keys. Database encryption is today a standard protection mechanism in most applications, such as the Active Directory (AD) service provided in a Windows server to handle things such as resources, clients, and permissions. Like most database applications, data managed by Active Directory encrypts data either at-rest or in-transit.

In general cryptography, symmetric and asymmetric encryption algorithms use key lengths or "bit-size" For example, Advanced Encryption Standard (AES) are symmetric keys that can vary in three different key lengths, 128, 192, and 256 bits. The key length determines the maximum number of combinations to overcome an encryption algorithm effectively. Naturally, a longer key length implies safer encryption. However, the key length could potentially impact database performance aspects such as throughput. This paper will explore how these variations in key lengths can impact database performance.

2.3.1 AES Encryption

Advanced Encryption Standard (2001), abbreviated AES, uses the Rijndael algorithm in combination with symmetrical block ciphers. AES is an algorithm that operates on bits of fixed lengths, i.e., blocks, and this algorithm encrypts plaintext data to an unintelligible form called ciphertext. It is used in symmetric-key cryptography, meaning that the same key is used both for encrypting and decrypting data.

AES is a more common encryption algorithm when securing data-at-rest and security for other aspects concerning databases (Samaraweera and Chang, 2019). Several proposed methods and frameworks are built around stopping attacks or safeguarding information using AES in some shape or form. Zaw, Thant, and Bezzateev (2019) investigated several methods to retrieve and securely store database data. The authors have implemented 256-bit AES encryption for row and column level encryption in one proposed method.

2.3.2 InnoDB Engine Encryption

This study will focus on general table encryption in MySQL and MariaDB using the InnoDB engine to secure data-at-rest, see figure 1. One of the more popular data-at-rest encryption algorithms in MySQL and MariaDB is the symmetric AES (Advanced Encryption Standard) with CBC (Cipher Block Chaining) mode. The AES algorithm supports 128, 192, and 256-bit key lengths. The InnoDB engine uses an encryption key to encrypt and decrypt selected tables.

After decryption, the user will be granted access to the tables to perform read and write operations.

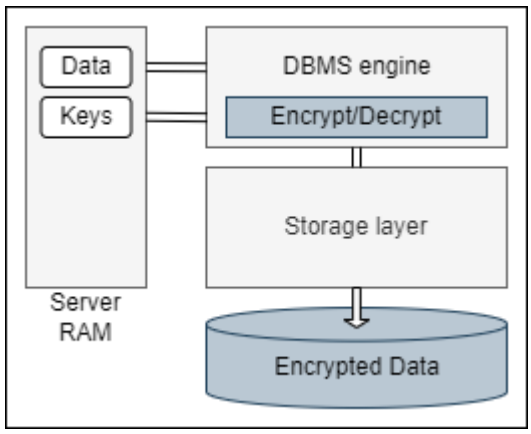

Figure 1 Example of InnoDB Encryption at The Storage Level

2.4 MySQL & MariaDB

MySQL is a SQL-based DBMS (Database Management System) first released in 1995. Similar to other RDBMS (Relational Database Management Systems), MySQL is built around row-based table structures that associate data elements between different tables. The MySQL DBMS is available in dual license, one for the community edition server under the GPL license, which will be used in this study, and a commercial license available to larger enterprises. MySQL has been further developed for a long time and provides many different functionalities. Some essential functionalities are the varying choices of encrypting data-at-rest, such as TDE (transparent data encryption), file-per-table, general tablespace encryption, and more (MySQL, no date).

MariaDB is a community-developed forked version of MySQL and was first released in 2009. As MariaDB is based on MySQL, many of the core functions of MySQL are also available in MariaDB. The main difference is that MariaDB is wholly open-source and operates entirely under the GPL License, contrary to MySQL, where features such as dynamic thread pooling are exclusive to the enterprise edition. Good thread pooling is essential to maintain high performance when the system faces an increasing number of clients connected to it. Leaving too many active threads open can lead to an unnecessary and costly context switching for the CPU. Dynamic thread pooling optimizes the number of threads needed to complete specific tasks and can grow or shrink the thread pool whenever required. Given these circumstances, the limitation of static thread pooling present in the community version of MySQL may directly affect the transactions per minute the database will be able to output when running the OLTP workload.

MariaDB primarily supports the same encryption functionalities that are available in MySQL. Even though MariaDB started as a fork of MySQL, both databases have been further developed in different directions. Many studies have been conducted when comparing various aspects of the databases, such as performance, replication, and scalability. One example of such a study is conducted by S.Tongkaw and A.Tongkaw (2016). The authors measured MySQL and

MariaDB's latest stable version at the time to uncover performance differences in terms of transactions per second (TPS), CPU, and memory usage using an OLTP workload. The authors concluded that the MySQL DBMS version had a comfortable performance lead over the MariaDB version.

The DBMS mentioned above has released newer versions since this article was published, and since technology evolves rapidly, the performance aspect may have changed during the years. This paper will explore the impact on performance in relational databases with the added overhead of encryption.

2.5 Performance Testing

Performance testing is the process of measuring and evaluating aspects related to the performance of a system (Jiang and Hassan, 2015). Performance testing can be applied to different system parts and an overall system test. Performance testing can also investigate the efficiency of algorithms or various configurations in a system.

Stress testing can be described as putting the system under immense continuous load for some time to measure the effectiveness of specific tasks, such as forcing the system to execute as many operations as possible. Load testing can be viewed as determining the behavior of a system during low, high, normal, or expected load to uncover load-related problems (Jiang and Hassan, 2015). Load-related problems can be functional like deadlocks, racing, or memory leaks. However, they can also be non-functional, e.g., quality or performance-related.

Jiang and Hassan (2015) also discuss the commonalities and differences among different testing techniques. The authors describe scenarios where these testing techniques are closely related in their survey. This study will focus on measuring the performance aspects of encrypted MySQL and MariaDB databases using load and stress testing to simulate a realistic use of the databases where virtual users are incrementally added to measure the effect of the encryption algorithms.

3 Research Scope

3.1 Problem Statement

The problem is the lack of studies measuring how TPM (transactions per minute) affects the performance of databases using AES (advanced encryption standard) data-at-rest encryption with an increasing number of concurrent users. The lack of thorough research on how specific data-at-rest encryption affects the throughput performance in different SQL databases where the same data-at-rest encryption algorithm is applied warrants further investigation.

3.2 Aim

This study investigates whether there is a significant difference in performance in terms of throughput when different AES versions are utilized in relational databases like MySQL and MariaDB, where the same technology for securing data-at-rest is employed. The database's transaction throughput is used to measure this performance. This study also investigates how the two databases compare against each other in terms of throughput when the AES-192 encryption is implemented.

3.3 Motivation

Several SaaS (software as a service) applications companies like Amazon, Microsoft, and Google use data-at-rest encryption (Saraswat and Tripathi, 2020). A benchmarking study comparing how different versions of the popular AES encryption algorithm would perform on other databases and the trade-offs would be helpful for developers looking to secure their data-at-rest.

Madyatmadja, Hakim, and Sembiring (2021) have conducted a study measuring the performance of implementing TDE (transparent data encryption) in an MSSQL database, capturing CPU time, memory usage, and TPM (transactions per minute). The study focuses on how the non-TDE versus TDE encrypted SQL database performed regarding reliability and efficiency. However, this research and other benchmarks do not specify which encryption algorithm TDE is implemented with or cover the available encryption algorithms used for data-at-rest encryption in-depth or the differences in performance between them.

Inspired by this and other studies, it would also be interesting to see how different SQL databases with data-at-rest encryption perform in terms of efficiency after applying various encryption algorithms.

3.4 Research Questions

RQ1: Is there a significant difference in transaction throughput between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL and MariaDB?

RQ2: Is there a significant difference in transaction throughput between MySQL and MariaDB when the AES-192 encryption algorithm is used to encrypt data-at-rest?

We aim to answer the above research questions with the two candidate SQL DBMS discussed previously in the background, namely MySQL and MariaDB.

3.5 Objectives

The objectives and individual responsibilities to fulfill the aim of this study are listed as follows:

1. Review related works, research DBMS and benchmarking procedures (Both).
2. Set up an adequate testing environment (Both).
3. Apply a proper configuration for MySQL server (Stewart).
4. Apply a proper configuration for MariaDB server (Mattias).
5. Specify the encryption algorithm to investigate (Both).
6. Configure the benchmarking tool in the testing environment (Both).
7. Measure TPM (transactions per minute) of different data-at-rest encryption algorithms (Individually done per assigned database).
8. Gather & analyze the results from the load tests (Both).
9. Measure NOPM (new order per minute) of the AES-192 data-at-rest encryption algorithm (Individually done per assigned database).
10. Gather & analyze the results from the stress tests (Both).

3.6 Hypothesis

As discussed earlier in the background section of this report, MariaDB is originally a community-developed forked version of MySQL. The core functionality is the same, and the databases are both based on the relational model and use SQL syntax. Both databases support similar "at-rest" and "in-transit" encryption algorithms. Even though MariaDB is a forked version of MySQL, the databases have further developed in different ways described earlier in the background section.

Considering these aspects, we have formulated a set of hypotheses that we want to investigate through our experiment with these aspects in mind. The hypotheses we test in this research are:

Ha_0 When measuring average transaction throughput, there is no significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MySQL.

Ha_1 When measuring average transaction throughput, there is a significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MySQL.

Hb_0 When measuring average transaction throughput, there is no significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MariaDB.

Hb_1 When measuring average transaction throughput, there is a significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MariaDB.

Hc_0 When measuring average transaction throughput, there is no significant difference between MariaDB and MySQL when the AES-192 encryption algorithm is used to encrypt data-at-rest.

Hc_1 When measuring average transaction throughput, there is a significant difference between MariaDB and MySQL when the AES-192 encryption algorithm is used to encrypt data-at-rest.

3.7 Division of Work

This subsection contains a simple table representation of who is responsible for which DBMS, see table 1. The responsibilities are also directly related to our objectives. A person sets up and configures a database and is also in charge of gathering results and analyzing that data.

Table 1 Division of responsibilities

Database	Stewart	Mattias
MySQL	X	
MariaDB		X

4 Related Work

There are some studies where the authors have performed benchmark experiments against different implementations of data-at-rest encryption. This section will introduce a few relevant articles related to our research. Some of the mentioned articles do not only investigate encryption in databases but also use some of the same tools and workloads to measure throughput that is present in this research. However, in contrast to this thesis, the related works do not specify which encryption algorithm they use when implementing data-at-rest encryption or reinforce their conclusions with any statistical evidence.

Madyatmadja, Hakim, and Sembiring (2021) conducted a benchmarking study comparing a non-TDE versus TDE (transparent data encryption) implementation in an SQL server. The authors aimed to measure the performance difference in the context of reliability and efficiency after implementing TDE. The authors used HammerDB as a benchmarking tool to perform an OLTP (online transaction processing) workload against the database. The metrics the authors chose to capture to compare the two implementations were CPU-time, memory management, and throughput with and without TDE.

The authors concluded that the performance degradation of implementing TDE was close to 7% in terms of reliability. Furthermore, the degradation would be up to 15% in terms of efficiency. However, the authors also concluded that the TDE implementation on the DBMS was still practical regardless of the degradation performance.

Natarajan and Shaik (2020) compare TDE implementations in different versions of Oracle DBMS, a relational model database. The study is focused on conducting a performance analysis to validate the claim that Oracle's latest DB version includes an improved TDE feature in optimizing CPU and lowering overhead at the storage level for processing data. The authors compare earlier DB versions of oracle with TDE implementation with the latest release, concentrating solely on CPU, RAM, and IO as metrics for their performance analysis and identification of possible bottlenecks.

The author's investigation concluded that Oracle's latest TDE feature in their new DB release had improved significantly. However, the authors also state that this new TDE feature does not provide details on improved RAM and IO performances. Their investigation shows that the optimization has positively affected CPU and IO. However, there is no significant improvement regarding RAM and encourages future work looking to improve this specific aspect in the TDE implementation.

Zhu et al. (2021) introduce a novel encryption mechanism called Full Encryption to Gauss DB. Gauss DB is a database of the relational model fully compatible with MySQL. According to the authors, the proposed implementation could protect data-at-rest and in-transit. The implementation uses two data encryption approaches at the column level, deterministic (DET) encryption and Randomized encryption (RND). DET means that the same plaintext results in the corresponding ciphertext, and RND generates a different ciphertext with the same plaintext.

The authors evaluate the performance in terms of average latency with insert, select, and update queries between unencrypted plaintext data and encrypted ciphertext in a few different scenarios. In one of the proposed scenarios regarding storing data in the cloud, the authors concluded that their full encryption implementation produced a comfortable performance

degradation. In this scenario, the proposed encryption solution introduced less than 5% overhead on the operations against ciphertext columns.

5 Methodology

This paper will investigate whether there is a difference in performance between relational databases when employing the same encryption algorithm for data-at-rest. To realize this, we chose a controlled experiment as our method. Wohlin et al. (2012) describe experiments as suitable for introducing different treatments in a technology-oriented approach. This choice of a method allows us to manipulate the independent variable in this experiment to observe the different outcomes while also exercising control over other variables that could potentially influence the results. Measuring the current level of control (the baseline) while evaluating other independent variable levels in a controlled environment will validate the measurements and produce accurate results that can be used for statistical analysis to perform adequate hypothesis testing.

5.1 Approach

Our research's first course of action was to explore how each DBMS operates and how data-at-rest encryption is implemented in the databases. This investigation led us to conduct initial research on how each DBMS operates and the necessary steps to enable data-at-rest encryption.

Research on suitable tools commenced, and the HammerDB benchmarking tool was chosen since it supports many relevant relational database management systems and has been used in other studies covered under the related works section. After we had settled on what benchmarking tool to use, we individually installed the database servers used in our experiment. The databases were configured to listen to the same TCP port so that one database could not run while another was active without manually disabling either of them.

By opting to conduct this study with MySQL and MariaDB as the two candidate RDBMSs, the main reasons behind this were their popularity, how they both operate under the GPL (General public license) license, and due to their compatibility with the chosen benchmarking tool, HammerDB.

Many different articles and documentation were examined to understand how a proper benchmark study is conducted and what crucial steps need to be considered to limit the number of possible validity threats, such as conclusion validity (Raasveldt et al., 2018).

One crucial factor Raasveldt et al. (2018) discusses when benchmarking DBMSs is the failure to optimize and sub-optimal configurations. The authors state that optimizing a DBMS for workloads is far from a trivial task, but steps should be taken to create as fair a performance comparison as possible. As MySQL and MariaDB use similar configuration files, we ensured that the configurations in both DBMSs were as similar as possible to ensure a fair performance comparison. The parameters we configured were the size of the buffer pool, concurrent threads, InnoDB log file size, and the number of encryption threads.

After installing the servers and configurations, we used the benchmarking tool to build the dataset into the databases. When creating the dataset, the benchmarking tool also created the stored procedures that the benchmarking tool uses to measure throughput, which will be later explained further in the experimental analysis section of this report. The benchmarking tool also explicitly creates indexes. Raasveldt et al. (2018) also discuss that indexes should either

be or not be created for the DBMS being benchmarked. Because the same dataset is generated in this experiment with both DBMSs, they will also have the same amount of indexes, ensuring fairness.

To ensure that each benchmark test is performed under as similar circumstances as possible, we ensured that each test performed in HammerDB was configured in the same manner. Each test was separately performed on the same physical machine with only the most relevant software installed.

For each database, MySQL and MariaDB, a series of load tests were performed. These load tests were done for each AES version as well as for one without any encryption enabled on the database. Each load test comprises five smaller sub-tests, each of which contains a different number of virtual users ranging from 5, 10, 15, 20, and 25 virtual users, respectively. In turn, each sub-test comprises five tests with the same amount of virtual users enabled to generate a broader data sample.

Essentially, each database undergoes four load tests, where each load test consists of 120 rows of data. Each of these rows of data is the per-row average TPM value across five subtests with different numbers of virtual users. Each of these subtests consists of the per-row average TPM value across five sample tests with the same amount of virtual users enabled. One complete load test consists of five subtests with five sample tests for each subtest. In short, 3000 rows of data for each load test.

These sample tests are configured to run an initial ramp-up phase of 1 minute. The virtual users connect to the database and start caching data in the database buffer, followed by a 3-minute test period. Configuring each test to run a 3-minute long test period while only collecting the first 2 minutes of test data was done due to how HammerDB tends to disconnect the virtual users before the 3-minute mark is reached, thus affecting the TPM value that is logged each second during the whole test. Since we are only interested in the TPM of when the test is working at peak performance, we discard the data collected during the ramp-up time and the last 60 seconds of each test as it does not contribute to a reliable result.

The data was later gathered and represented in box plots to visualize better how each AES encryption performed during the load tests. This representation is also suitable for visualizing outliers and signs of potential skew in the measured data for each sample. After the results were gathered and presented, an analysis was performed with an ANOVA test in order to either accept or reject the initial null hypothesis.

We were curious if a performance comparison between the two databases could be made based on the results. However, a comparison between databases based on the TPM metric might lead to incorrect assumptions due to a limitation in the benchmarking tool. Although, if we want to compare databases against each other, another metric known as NOPM (New orders per minute) has to be taken into account. Further explanation of the technical aspects of these metrics is covered in the experimental analysis section of this report.

We decided to conduct a series of stress tests with the NOPM metric. We performed these tests with the AES-192 encryption algorithm applied to both databases due to how the initial data gathered from the load tests indicated a high transaction throughput on MySQL with this specific AES encryption. We later gathered and analyzed the results from these stress tests. An

analysis was performed with an ANOVA test in order to either accept or reject the initial null hypothesis.

As NOPM is measured during the entire test with a fixed value being reported at the end of each test, actively collecting this value during the load tests together with TPM would lead to misleading results. This is due to the choice we made of only collecting the data results from the first two minutes of the three-minute test period, meaning that the NOPM value would be calculated based on data not used in the final dataset.

5.2 Alternative Methods

These alternative methods could have been used in different settings, and why we considered them but ultimately did not choose them.

One alternative method that we could have conducted is a case study. We did not conduct such a study because it would be challenging to generalize differences in encryption algorithms across relational databases. The findings would only apply to the databases chosen to be a part of the study. Wohlin et al. (2012) describe case studies as observational with a lower level of control than an experiment and are used to explain an undiscovered phenomenon instead, which is not the case with DBMSs since extensive research already exists. In the context of a case study, it would be difficult with the lack of research regarding benchmarking of data-at-rest encryptions to support the conclusions with clear statistical significance.

A survey is another alternative method. Gathering qualitative or quantitative data with a survey in this context could be done by interviews or questionnaires. Considering the time constraints, a problem with this method is that it would be difficult to find enough experts on data-at-rest encryptions with different databases to conduct an interview. On the other hand, a questionnaire would be possible, but we can not realistically predict the response rate or find enough experts on the subjects to send a questionnaire. In the case of a low response rate, the results might not be generalizable to a broader population and would undermine our research's external validity.

5.3 Dependent and Independent Variables

This experiment aims to determine if there is a significant difference in performance between the chosen databases, and to measure this, the following dependent and independent variables have been identified.

The main independent variables identified in the first experiment are the different AES versions with different key lengths, along with the different amount of virtual users used during the load tests. One could also argue that the databases themselves are the main independent variables during the second experiment as the AES encryption and the number of virtual users stayed the same throughout the stress tests. Our dependent variables are the transactions per minute a database can perform before and after applying different encryption algorithms, and the new orders per minute a database can perform when a specific AES encryption is applied. According to Wohlin et al. (2012), there is often only one dependent variable in an experiment. In agreeance with this, it would thus be sufficient to determine if a significant difference exists when measuring performance by using these throughput metrics in each experiment respectively.

6 Experiment Analysis

6.1 System Specification

The system specifications on which the DBMS and HammerDB were set up are illustrated in Table 2:

Table 2 System Specification

OS	Windows 10 Pro
CPU	Intel Core i9-9880H 2.30GHz, 2304 MHz, 8 Cores, 16 Logical Processors
Memory	32GB RAM
Disk	NVMe KIOXIA 512GB SSD

6.2 HammerDB & TPC-C Workload

The tool used for benchmarking the performance is the open-source HammerDB benchmarking tool. HammerDB is used to load test and benchmark databases by simulating workloads done by a set of virtual users against a specific database. This tool derives valuable information about differences in performance depending on hardware and software configurations. The OLTP workload in the TPROC-C schema used in this study is an open-source workload derived from the TPC-C Benchmark Standard. The advantage of this workload is that it is independent of any specific database implementations or hardware configurations. Additional benefits of this benchmark protocol are its scalability and simulation of the transaction flow between users and a database in a real-world industry setting. This benchmark protocol also allows measuring performance in new-order transactions per minute and the overall transaction rate measured in TPM (Transactions Per Minute). The TPC-C protocol implements a system to simulate orders from customers to a company. The company keeps its stock in warehouses where each warehouse has a set of sales districts, and each district supplies a set of customers. See figure 2 for reference.

Each workload consists of a mix of five transactions selected at random, where these transactions vary between Read, Write, and Update queries. The following stored procedures are listed below, and all of them make use of Read, Write and Update in some shape or form:

- New Order
- Payment
- Delivery
- Order-Status
- Stock-Level

The listed transactions are executed by stored procedures generated by the benchmarking tool in combination with building the warehouse schema. The TPROC-C standard used by HammerDB can not be used to compare with official TPC-C results due to how it only complies

with a subset of the TPC-C Benchmark Standard. This experiment should therefore not be considered equal to an actual TPC-C test. More information about these standards is available on the HammerDB document page.

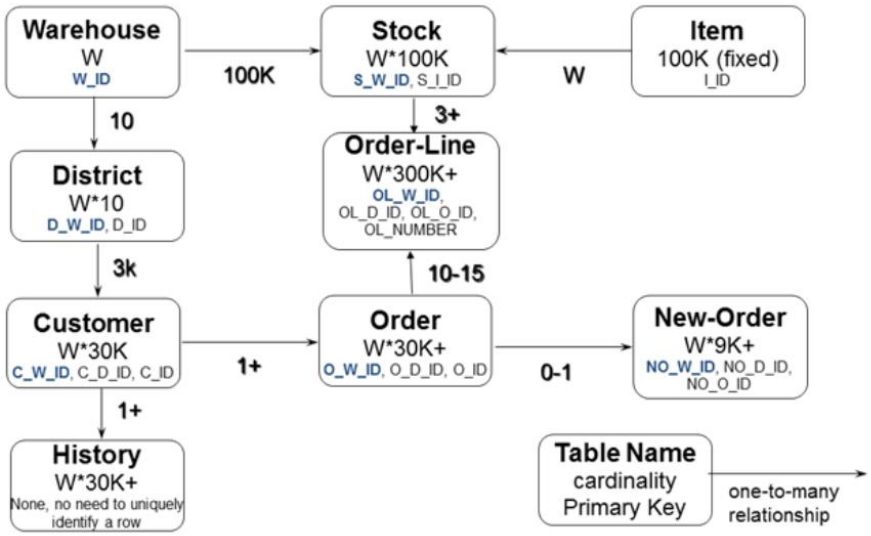

Figure 2 TPROC-C test schema

6.3 HammerDB Configuration and Metrics

For our experiment, we chose HammerDB version 4.3 and configured the TPROC-C schema with 100 warehouses. To simulate a load testing environment, we gathered the data measured in TPM in incremental steps detailed in the approach section of this report. Each load test was performed on a database with and without a specific AES encryption enabled.

During the stress test part of this experiment, we configured HammerDB to log NOPM (New order per minute) that was used to compare the throughput rate between the chosen SQL databases. NOPM is a cross-database comparable metric generated by the testing tool independent of any specific database implementation. Since the TPM metric records all transactions, whether they are commits or rollbacks during the measuring period, it does not necessarily adhere to the ACID principles described earlier in this report when logging the transaction rate. However, NOPM does not have this limitation and can therefore be used to perform a fairer comparison between two different databases.

6.4 Results

In the results, each load test consists of five smaller tests where the first test utilizes a set of 5 virtual users which then increases throughout the load test to a peak of 25 virtual users in the last of the smaller sub-tests. These smaller sub-tests are individually the average sum across five identical tests. This was done because having a larger sample size will yield a more accurate result, thus making each box in the boxplot a more accurate representation of the TPM distribution during the workload.

6.4.1 MySQL Unencrypted

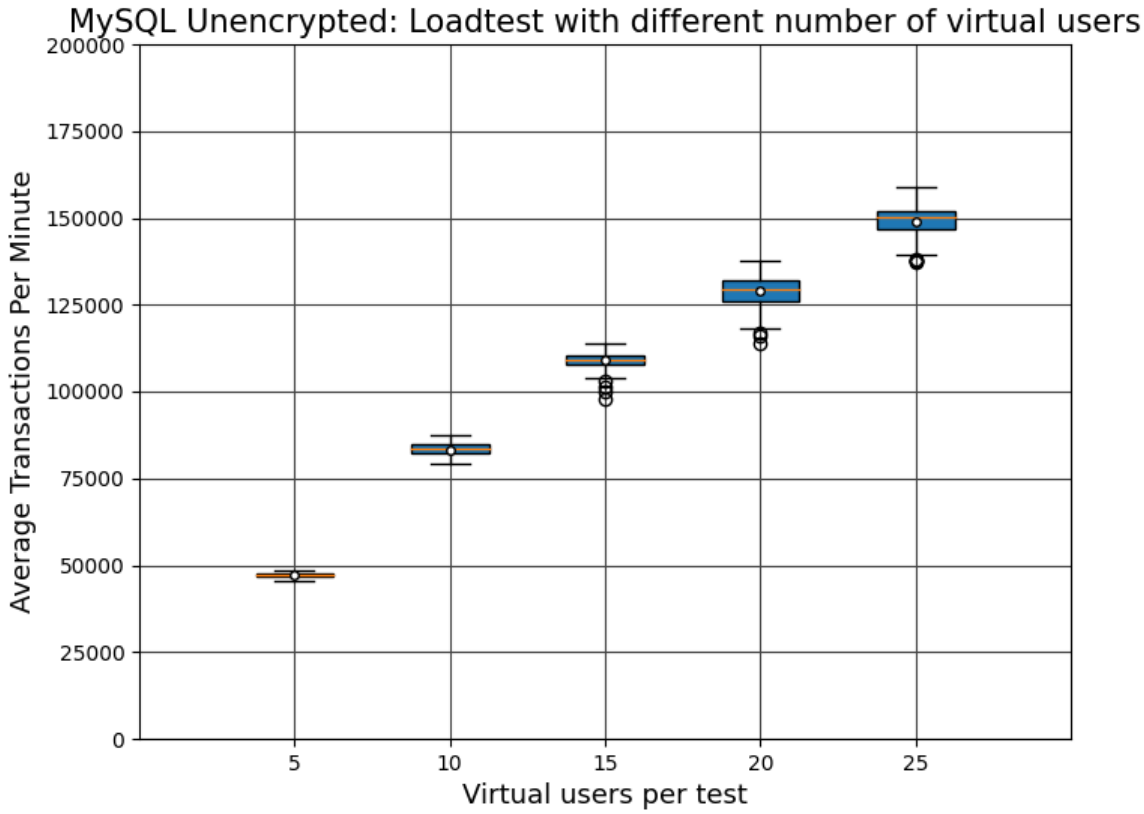

Figure 3 MySQL load test on an unencrypted database

Figure 3 shows the load test on the unencrypted MySQL schema, where the TPM ranges between 25k to 200k. There is no indication of any outliers during a lower load with 5 and 10 virtual users. The interquartile range in the graphs is relatively small, indicating that the data is reliable. The outliers become apparent during medium to higher load, where the sample consists of 15, 20, and 25 virtual users. However, these outliers are not extreme and are most likely due to how the DBMS handles the transactions. The performance seems to degrade as more users are running the workload, especially in the higher load range of 20 to 25 virtual users.

6.4.2 MySQL AES-128 Encryption

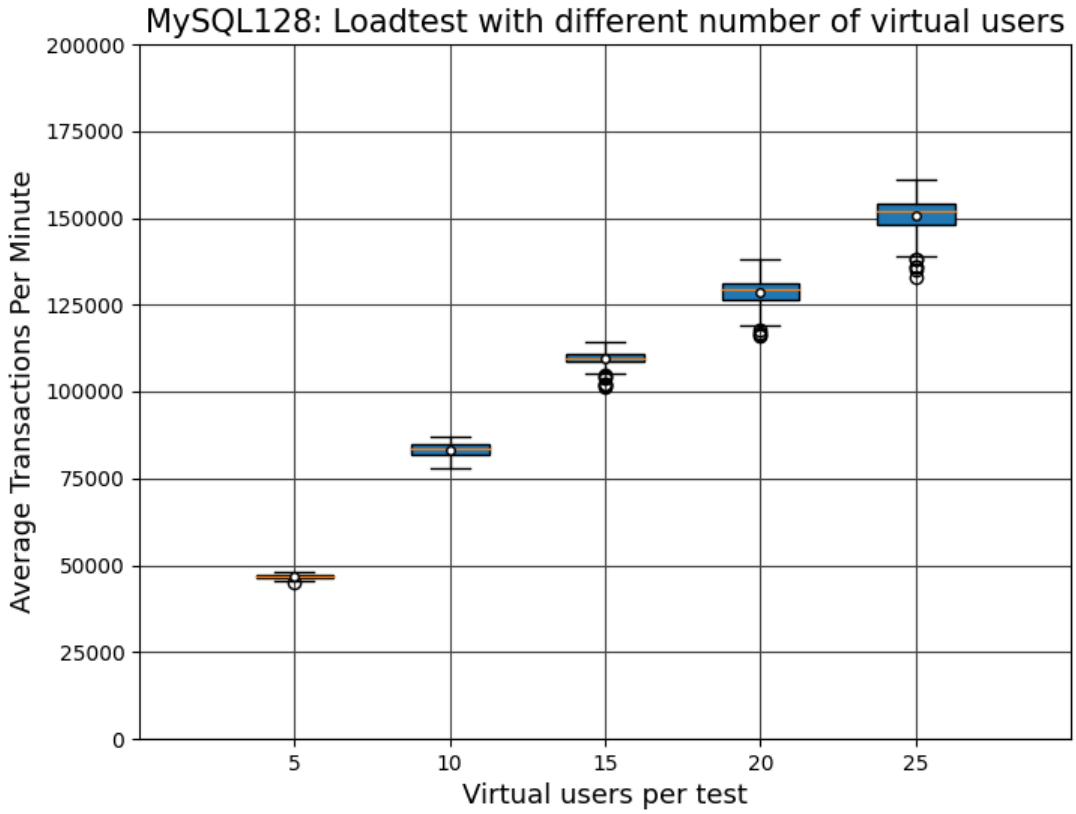

Figure 4 MySQL load test on a database encrypted with the AES-128 algorithm

Figure 4 shows the load test on the MySQL schema with AES-128 encryption enabled on the dataset. The interquartile is still relatively small, as previously visualized in figure 1, indicating that the data sampling is good. The outliers seem apparent during low load with five virtual users, but these are few and close to the minimum value, so they should not significantly affect that data sample.

6.4.3 MySQL AES-192 Encryption

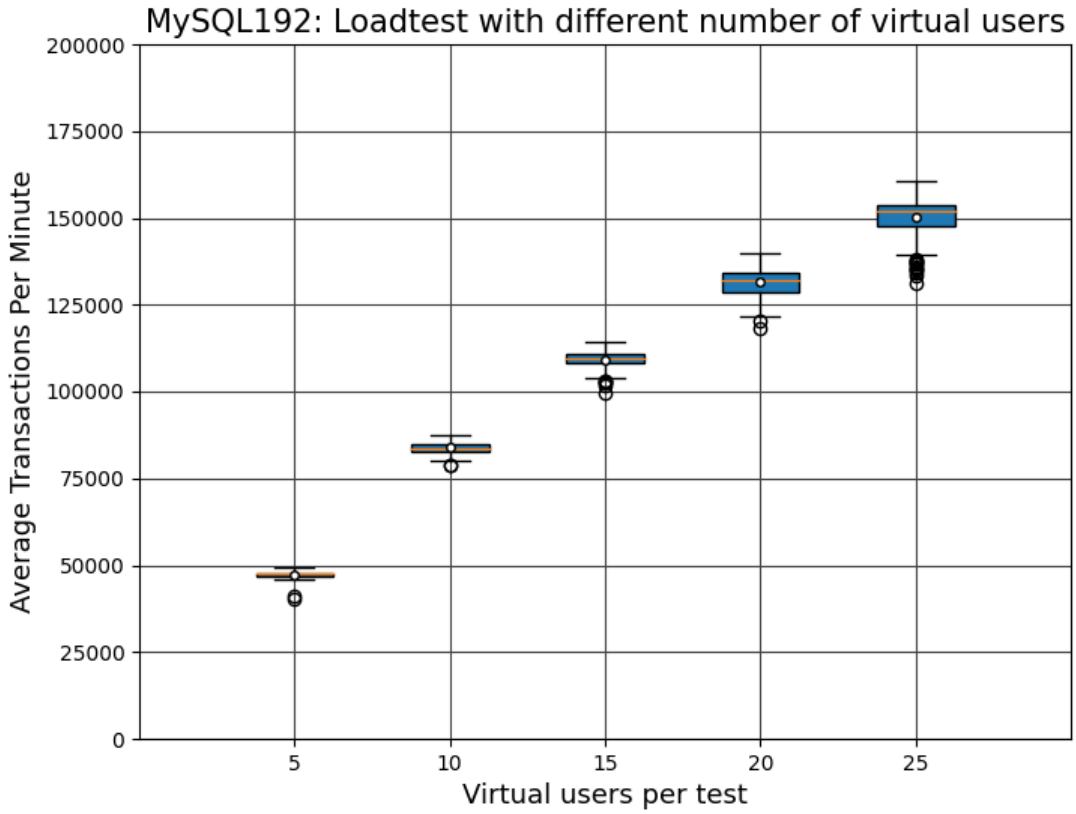

Figure 5 MySQL load test on a database encrypted with the AES-192 algorithm

Figure 5 shows the load test on the MySQL schema with AES-192 encryption enabled on the dataset. The interquartile sizes are similar to previous results, and the increased load seems to be following the same pattern as the previously shown results. As opposed to the previous AES-128 encryption, the outliers in these samples become more apparent in all virtual users per test. During the highest load, the outliers seem to be higher in quantity, but the rate of transactions is still in the same general range as in the other figures.

6.4.4 MySQL AES-256 Encryption

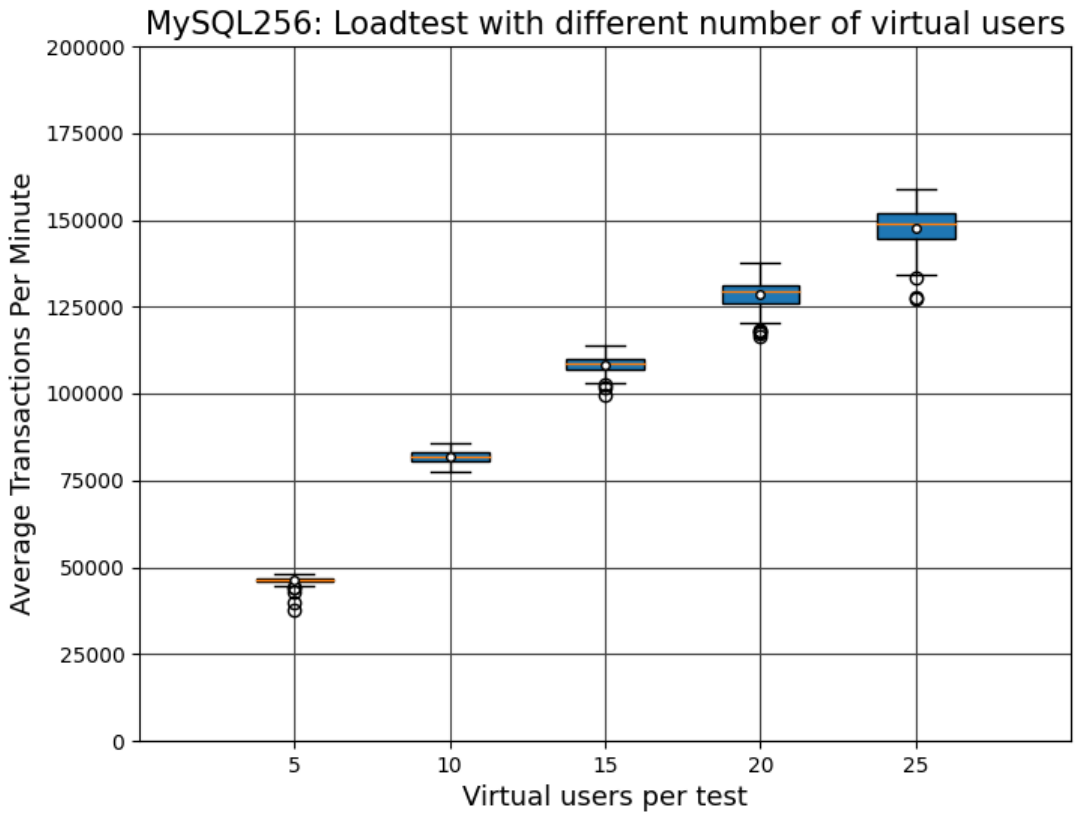

Figure 6 MySQL load test on a database encrypted with the AES-256 algorithm

Figure 6 shows the load test on the MySQL schema with AES-256 encryption enabled on the dataset. The results generated with this encryption algorithm look very similar to the other results presented previously and follow the same pattern whereby as the number of virtual users increases by five, the average amount of transactions does not increase linearly. There are also some minor differences in how this non-linear growth is manifested, as the TPM for each set of virtual users does not perfectly overlap with the results from the previously tested AES key lengths.

6.4.5 MariaDB Unencrypted

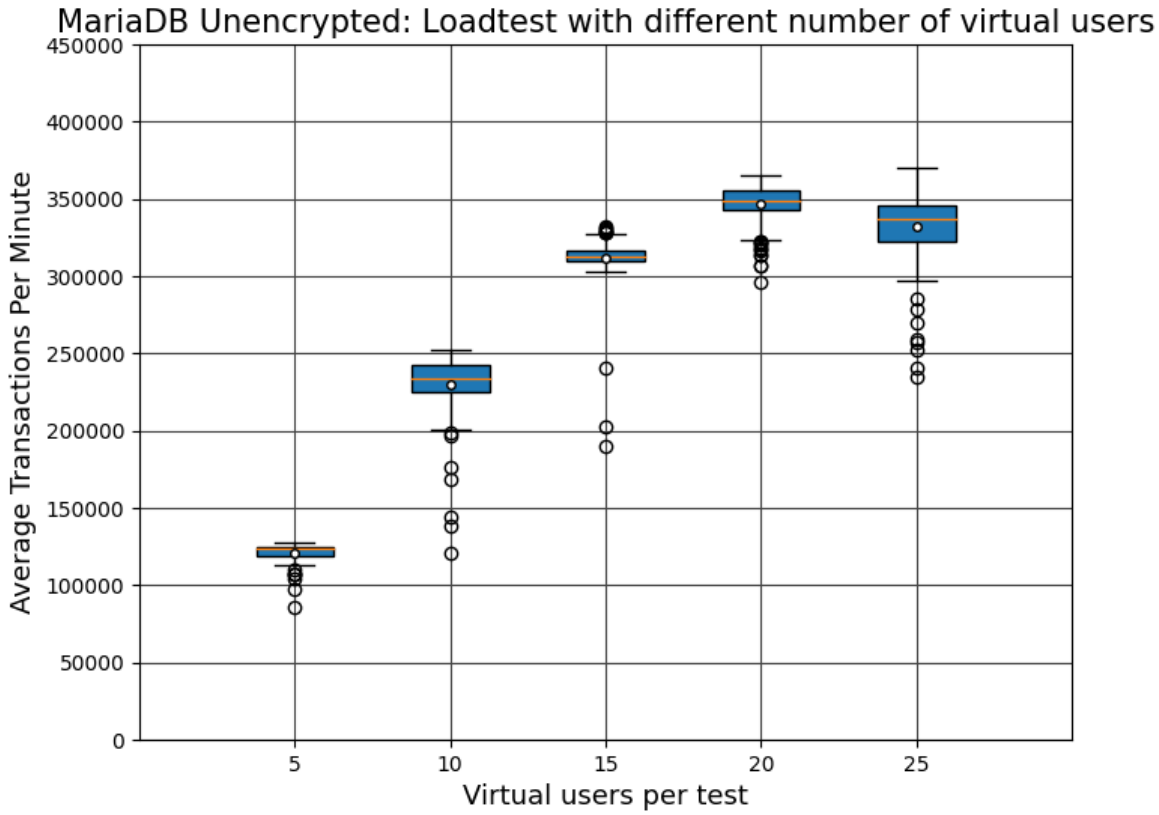

Figure 7 Load test on an unencrypted MariaDB database

Figure 7 shows the load test on the unencrypted MariaDB schema, where the TPM ranges from 100k to 400k. The results display an initial somewhat linear growth between the first three sets and a slight drop in TPM when running 25 virtual users. The number of outliers also seems to vary significantly between each test set.

6.4.6 MariaDB AES-128 Encryption

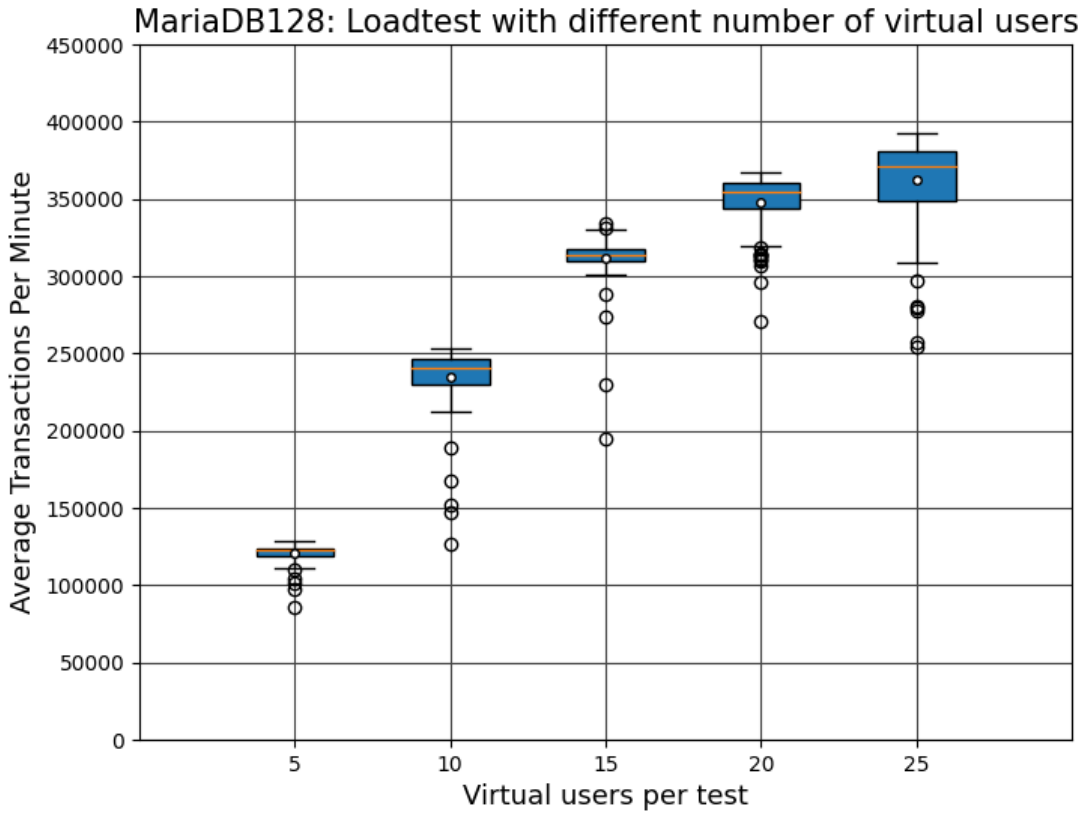

Figure 8 Load test on a MariaDB database encrypted with the AES-128 algorithm

Figure 8 shows the load test on the MariaDB schema with AES-128 encryption enabled on the database. The initial growth in average TPM over the first three sets, followed by somewhat stunted growth in the last two sets, seems to align with the previous data gathered from the test with the unencrypted MariaDB database. The decline in TPM on the previous dataset with 25 virtual users does not seem to be as pronounced as on the previously unencrypted load test in MariaDB.

6.4.7 MariaDB AES-192 Encryption

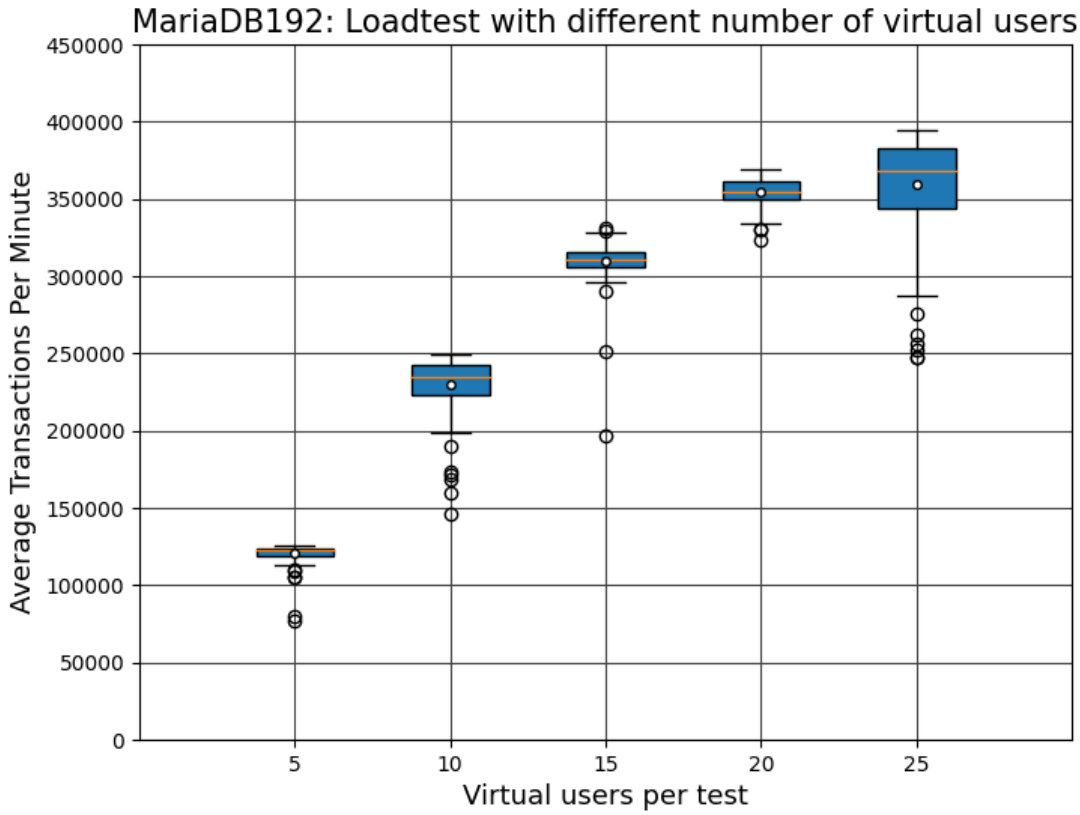

Figure 9 Load test on a MariaDB database encrypted with the AES-192 algorithm

Figure 9 shows the load test on the MariaDB schema with AES-192 encryption enabled on the database. Apart from slight variations, the data collected from this load test seem to be very similar to the previous load test with the AES-128 encryption in MariaDB.

6.4.8 MariaDB AES-256 Encryption

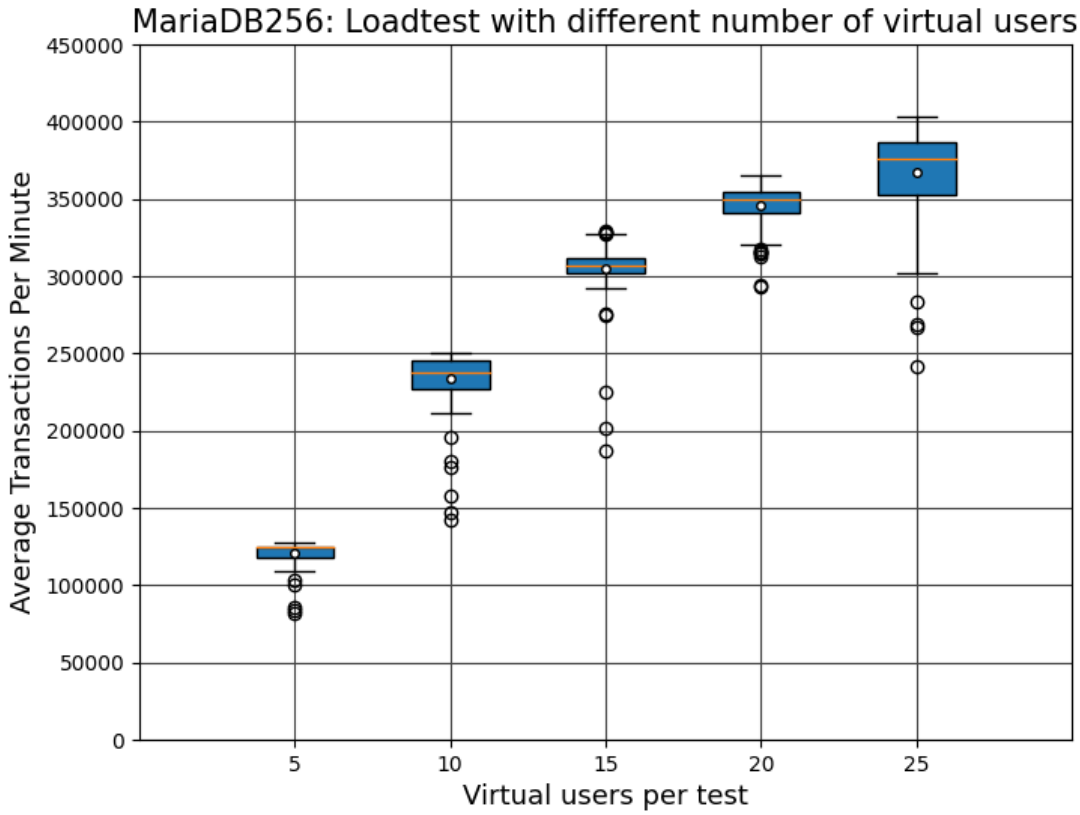

Figure 10 Load test on a MariaDB database encrypted with the AES-256 algorithm

Figure 10 shows the load test on the MariaDB schema with AES-256 encryption enabled on the database. The data gathered from this load test seem to be quite similar to the previous AES-192 load test data. However, the regression in TPM that seems to occur across each set of virtual users in the previous load tests done with MariaDB does not seem to be as significant in the AES-256 load test.

6.5 Algorithm Comparison

An ANOVA test is initially performed for each database with the different encryption algorithms to conduct the statistical analysis. If a significant difference can be statistically proven, a Tuckey test will follow it. This potential difference will then later be compared to the baseline test. This approach aimed to confirm with a 95 percent confidence level if there were any significant differences in performance between the encryption algorithms in the relational databases used in this experiment.

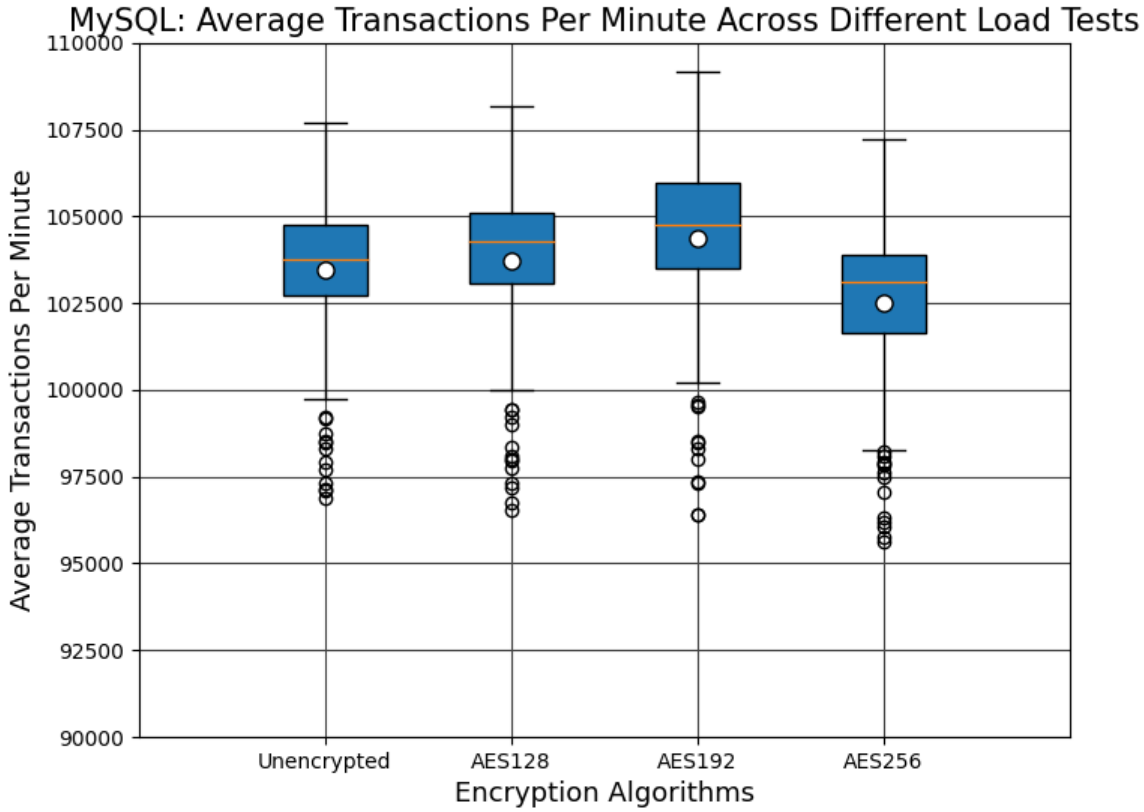

Figure 11 Comparison between different encryption algorithms in MySQL

Table 3 MySQL: Standard Deviation and Mean Value

Algorithm	Standard Deviation	Mean Value
Unencrypted	2370,9	103445,3
AES-128	2553,4	103729,8
AES-192	2720,2	104371,2
AES-256	2616,4	102498,6

Figure 11 displays the total average TPM distributions across the load tests done on MySQL. The line in the boxplots represents the measurement median value, and the white circle represents the mean value. Table 3 consists of the standard deviations and mean values for the algorithm comparisons on MySQL.

A variance in average TPM is observed between the encryption algorithms. The number of outliers also seems to be reasonably consistent across all load tests. By conducting an ANOVA test between all three encryption algorithm load test results, the ANOVA test concluded with a P-value of 0.0002915. A 95 percent confidence level strongly indicates a significant difference in performance between the encryption algorithms. A Tukey test determined which algorithm contributed to the statistical significance. There was no significant difference between the AES-128 and AES-192 algorithms, with a p-value of 0.1435. The considerable difference originated between the AES-128 and AES-256 algorithms with a p-value of 0.0009639 and between AES-192 and AES-256 with a p-value of 0.0000002027. The AES-192 and AES-256 pair have the most significant discrepancy in the Tukey test.

Comparing the encryption algorithms to the baseline measured without any data-at-rest encryption implemented indicates no disadvantage in transaction throughput when adding data-at-rest encryption except for the AES-256 algorithm, which seems to produce a lower range of average TPM. The ANOVA test done on the results from all the load tests on MySQL in figure 11 confirms that there exists a significant difference with a p-value of 0.0000005. The Tukey test confirmed a significant difference between the unencrypted baseline and AES-192 with a p-value of 0.0278 and between the unencrypted baseline and the AES-256 with a p-value of 0.02319.

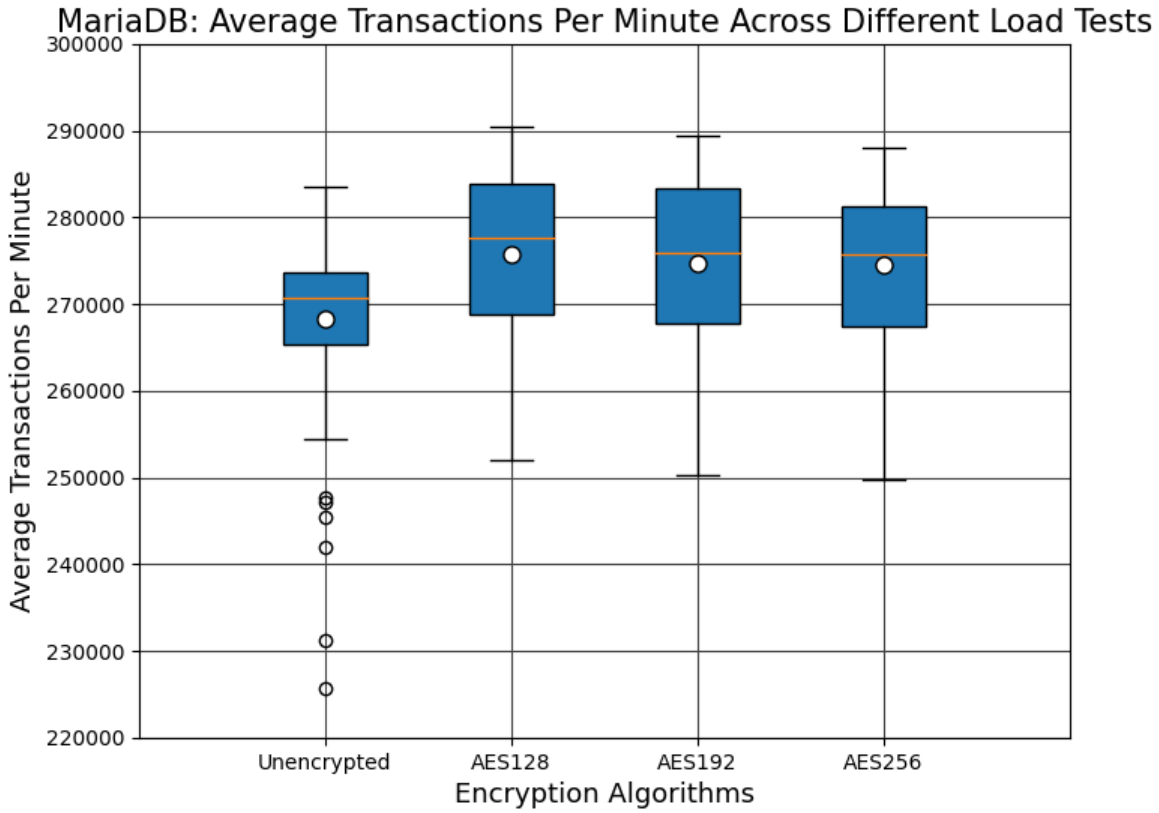

Figure 12 Comparison between different encryption algorithms in MariaDB

Table 4 MariaDB: Standard Deviation and Mean Value

Algorithm	Standard Deviation	Mean Value
Unencrypted	9288,2	268261,0
AES-128	9701,1	275717,3
AES-192	9464,6	274798,3
AES-256	8944,7	274501,0

Figure 12 displays TPM distributions across the load tests done on MariaDB. The boxplot line represents the measurement's median value, and the white circle is the mean value. Table 4 consists of the standard deviations and mean values for the algorithm comparisons on MariaDB.

A clear distinction can be made from the diagram as the unencrypted version seems to perform slightly worse on average, similar to the observation done with MySQL results. Another interesting observation is the lack of outlying values on either of the three AES-key lengths. By conducting an ANOVA test between all three encryption algorithms, the ANOVA test concluded with a P-value of 0.5781. A 95 percent confidence level broadly indicates no

significant difference in performance between the encryption algorithms. Thus, the different key lengths do not seem to impact transactional performance significantly.

In contrast to the load tests done with MySQL, the load tests done with MariaDB indicate no loss of transactional throughput when utilizing any of the AES data-at-rest encryption algorithms over the unencrypted one. Instead, the load tests done with any AES encryption enabled seem to outperform those without any encryption enabled.

The ANOVA test conducted on the results from the load tests on MariaDB in figure 12 confirms that there exists a significant difference with a p-value of 0.0000000007. A Tukey test confirms that there is a significant difference between the baseline test and the AES encryption algorithms with a p-value of 0.000000008712, 0.0000005868, and 0.000002087, respectively, in the order presented in figure 12. This discrepancy between the unencrypted and encrypted versions seems to correlate with the findings in related research by Madyatmadja, Hakim, and Sembiring (2021).

6.6 Database Performance Comparison

As mentioned in section 6.3, comparing the transactional throughput between two databases in HammerDB can not be done solely based on the number of TPM produced. Due to this, a comparison done based on the NOPM (New orders per minute) metric is instead the recommended approach when comparing transactional throughput between two databases.

An analytic comparison between the two databases was performed between MariaDB and MySQL by first conducting six stress tests with 30 virtual users enabled for each test on MariaDB and MySQL with the AES-192 encryption enabled. The reason for conducting the tests with 30 virtual users was how this would pose a sufficient level of stress based on the results from the previous load tests performed on each of the two databases. The results from the load tests done on MySQL in figure 11 indicated that the highest levels of TPM were reached with the AES-192 encryption enabled. The choice to conduct the stress tests with the AES-192 algorithm was considered optimal since no significant difference was found between the different AES versions on MariaDB and how the AES-192 displayed the highest throughput on MySQL.

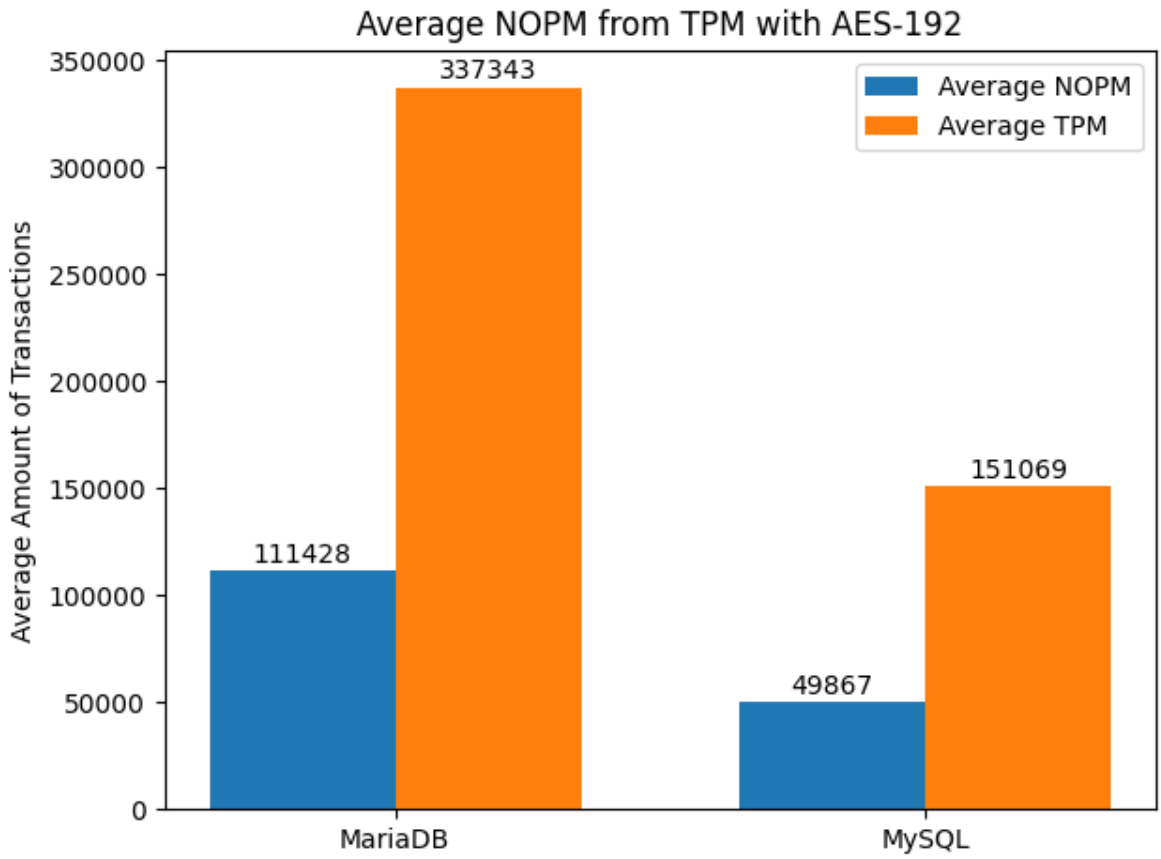

Figure 13 Performance Comparison between MariaDB and MySQL with AES-192 Encryption

Figure 13 displays the number of NOPM (New orders per minute) and TPM (Transactions per minute) achieved as averages across six stress tests conducted on MariaDB and MySQL. The number of new orders per minute is roughly 33% of the total amount of TPM from the stress

tests done with both databases. The distribution of approximately 1/3 of NOPM per total TPM in both databases. If this was not the case, it could indicate potential errors, according to the authors of the benchmarking tool (Hammerdb, 2018).

Given the results from the performance comparison, MariaDB vastly outperforms MySQL in both TPM and NOPM. These results also reflect the difference in average transaction values presented in the load tests. Since the NOPM value is independent of any specific database implementation, as explained in the methodology section of this report. We can confidently state that MariaDB processes transactions at a rate two times higher than MySQL. While the difference is visually clear in figure 13, an ANOVA test confirms this significant difference with a confidence level of 95% with a p-value of 0,00000000000002.

6.7 Conclusions

6.7.1 Testing the Hypotheses

Six hypotheses are stated in the problem section (3.6 Hypothesis). The first four (Ha_0, Ha_1, Hb_0, Hb_1) revolve around possible performance differences between the AES encryption algorithms used in each database. The remaining two (Hc_0, Hc_1) revolve around possible performance differences between each database when the AES-192 encryption algorithm is applied. The following conclusions were drawn based on a statistical analysis that, through ANOVA tests, verifies if a significant difference exists and, through Tukey tests, determines where these differences lie.

Hypothesis Ha_0 state that:

When measuring average transaction throughput, there is no significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MySQL.

The results of the ANOVA test prove with a confidence level of 95% that there is a significant difference between the investigated AES algorithms when implemented in MySQL. The Tukey test shows which algorithms contribute to this significance and that the AES-192 encryption produces the most significant difference in transaction throughput. This significance can be observed in figure 11 and proven by the P-value result generated with the Tukey test. The results presented in figure 11 show that AES-192 performs better in terms of average TPM than AES-256. However, if security is a priority, we would recommend adopting the most secure AES algorithm when implementing data-at-rest encryption if the TPM (transactions per minute) is similar or not much different from the MySQL algorithm comparisons.

Based on the results from the statistical analysis, we strongly suggest that the null hypothesis Ha_0 should be rejected. With this statistical evidence, we accept the alternative hypothesis Ha_1 that state: *When measuring average transaction throughput, there is a significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MySQL.*

Hypothesis Hb_0 state that:

When measuring average transaction throughput, there is no significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MariaDB.

The results of the ANOVA test prove with a confidence level of 95% that there is no significant difference between the investigated AES algorithms when implemented in MariaDB. This insignificant difference in means is statistically proven by the P-value of 0.5781 generated by the ANOVA test. As shown in figure 12, the transaction throughput is relatively similar across all algorithms observed, ranging from about 250 000 TPM to 290 000 TPM. Based on these results in figure 12, we can conclude that different key lengths do not matter for the AES algorithm in MariaDB for this experiment. We recommend using the most secure AES algorithm for implementing data-at-rest encryption since the transactional throughput is similar across all investigated AES versions.

Based on the results from the statistical analysis, we strongly suggest that the null hypothesis Hb_0 should be accepted, thus rejecting the alternative hypothesis Hb_1 that state: *When measuring average transaction throughput, there is a significant difference between data-at-rest encryption algorithms AES-128, AES-192, and AES-256 in MariaDB.*

Hypothesis Hc_0 state that:

When measuring average transaction throughput, there is no significant difference between MariaDB and MySQL when the AES-192 encryption algorithm is used to encrypt data-at-rest.

The results of the ANOVA tests on the stress testing data prove with a confidence level of 95% that there is a significant difference in transactional throughput between MySQL and MariaDB. This difference was initially observed in the load testing figures of MySQL and MariaDB, where the TPM value was a lot higher when running the workload against the MariaDB database than with the MySQL database. For reasons explained earlier in the methodology section of this report, the NOPM (new orders per minute) metric in relation to TPM (transactions per minute) confirmed previous observations that MariaDB does outperform MySQL when running the same workload as a stress test with data-at-rest encryption implemented.

Based on the results from the statistical analysis, we strongly suggest that the null hypothesis Hc_0 should be rejected. With this statistical evidence, we accept the alternative hypothesis Hc_1 which states: *When measuring average transaction throughput, there is a significant difference between MariaDB and MySQL when the AES-192 encryption algorithm is used to encrypt data-at-rest.*

6.7.2 Answering the Research Questions

The first research question we intend to answer states:

Is there a significant difference in transaction throughput between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL and MariaDB?

The investigated encryption algorithms in MariaDB barely differ since they seem to fall into the same TPM range of 250 000 to 290 000. In contrast, a more evident difference between the compared algorithms in MySQL exists, as the worst-performing algorithm has a considerably lower TPM range compared to the other encryption algorithms implemented in the MySQL database. These results, on the surface, indicate a substantial difference in transaction throughput based on the gathered data from the measured average transactions per minute.

The findings from the statistical analysis done on the results from the tests conducted on MariaDB and MySQL, which were performed with ANOVA tests and followed by Tukey tests where it was needed, concluded with a P-value greater than 0.05 for MariaDB and P-values below 0.05 for MySQL. These results show that a significant difference in transaction throughput between different AES encryption algorithms in relational SQL databases does exist. In conclusion, there is a significant difference in transaction throughput between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL but not in MariaDB.

The second research question we intend to answer states:

Is there a significant difference in transaction throughput between MySQL and MariaDB when the AES-192 encryption algorithm is used to encrypt data-at-rest?

The results indicate a significant difference between MariaDB and MySQL when the AES-192 encryption algorithm is applied. The average NOPM (new orders per minute) values reported from the stress tests done with MariaDB far exceed the values reported from the stress tests done with MySQL, which can be clearly observed in figure 13, where MariaDB produced an average total NOPM of 111428 and MySQL produced an average total NOPM of 49867. Given the results, it is proven that a significant difference in terms of performance exists between MySQL and MariaDB, where MariaDB performs at least twice as well as MySQL when processing transactions.

In this experiment, regardless of the encryption algorithm, MariaDB will always perform better when handling transactions, contrary to the findings by S.Tongkaw and A.Tongkaw (2016), where MySQL performed better.

7 Discussion

7.1 Summary and Discussion

7.1.1 Summary

This research aims to compare different key lengths of the AES encryption algorithm used to encrypt data-at-rest in relational databases and measure the impact on performance in terms of transaction throughput. This research also aims to measure how two relational databases compare in terms of transaction throughput when the same AES key length is applied.

The first research question stated: *"Is there a significant difference in transaction throughput between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL and MariaDB?"*

Data-at-rest encryption in the selected databases was implemented with different versions of the AES encryption algorithm with varying key lengths. A benchmarking test was then performed, followed by a statistical analysis of the results, which concluded that a significant difference exists between the AES-128, AES-192, and AES-256 algorithms used to encrypt data-at-rest in MySQL but that no significant difference between these exists when tested in MariaDB. In the larger context of relational databases, our findings align with the results of similar related works in this research area. Based on our findings, our answer to the first research question is that there is a significant difference in transaction throughput between the different AES key lengths in the two relational SQL databases, with the difference located between the AES key lengths used in MySQL.

The second research question stated: *"Is there a significant difference in transaction throughput between MySQL and MariaDB when the AES-192 encryption algorithm is used to encrypt data-at-rest?"*

The AES-192 encryption algorithm used to encrypt data-at-rest was implemented in both of the selected databases. Each database was subject to stress tests after which the results underwent a statistical analysis that confirmed the existence of a significant difference in terms of transaction throughput between the two databases. Based on our findings, our answer to the second research question is that there is a significant difference in transaction throughput between MySQL and MariaDB when AES-192 is applied to encrypt data-at-rest.

7.1.2 General Discussion

To answer the aim of this study, a set of objectives were put together that are covered in section 3.5 under the research scope of this thesis.

To fulfill the first objective, articles relating to data-at-rest encryptions in relational databases were investigated. These articles led us to determine which relational databases provided mechanisms and support for enabling data-at-rest encryption. The DBMSs initially chosen along with MySQL and MariaDB were MSSQL (Microsoft SQL) and OracleDB (Oracle DBMS). However, we discovered that the licensing agreement of MSSQL and OracleDB did not allow the publication of benchmarks without the company's consent. Since we did not have the time to communicate our request to the companies, we opted for only including MariaDB and MySQL in this study since they are under a GPL (General public license) license, and the

results would be free to publish. The benchmarking tool HammerDB was chosen for the experiment in this study because it supported our selected databases and has automated functionality for load testing. The benchmarking tool is also used in other related studies, such as the benchmarking study by Madyatmadja, Hakim, and Sembiring (2021).

To fulfill the second objective, we decided to borrow a computer from the company at which we were writing our thesis. This computer came with a fresh installation of the operating system (Windows). We only used it to download the necessary servers and software to run the DBMSs and the benchmarking tool. We decided to do this because we wanted to limit the possibility of other processes running that could affect the internal validity of our experiment.

To fulfill objectives 3 and 4, we researched MySQL and MariaDB to determine which parameters can be configured for the benchmarking tool to function consistently. We found that there are no one-size-fits-all solutions for database optimization. However, we followed some guidelines provided by the tool's documentation and agreed upon a configuration that we individually applied to the respective databases to keep them consistent and under the same conditions. This configuration consists of tuning both databases buffer pool, concurrent threads, InnoDB log file size, and encryption threads parameters. While applying this configuration, we noticed a performance gain in both databases when running a series of pilot tests.

Fulfilling the fifth objective involved researching the different encryption mechanisms and supported algorithms for data-at-rest encryption in MariaDB and MySQL. There were several ways of implementing data-at-rest, such as column-level encryption or table-level encryption. In this study, we opted for table-level encryption using the InnoDB engine. The only algorithms supported in MySQL and MariaDB at writing were the AES encryption algorithm with varying key lengths. Since they lacked general support for other encryption algorithms, we included all versions of the AES algorithm in our research.

Fulfilling the sixth objective involved configuring the benchmarking tool to build the dataset into the databases and managing the workload. Issues were apparent when testing this tool, such as that the TPM counter recorded null values before the test began as well as data from when the test ramped up and ramped down. This issue could be handled by manual adjustments to the data and the testing configuration as described in the approach section of this report.

Fulfilling the seventh objective involved performing the load tests with each database using each AES key length. The cached data was cleared before each test, and during each test, the TPM value was logged and stored for further analysis.

The eighth objective consists of collecting the data result for each load test and preparing it for analysis. To better visualize the results, we decided to use box plots since other diagram types, such as histograms or bar charts, would not allow us to view the distribution as adequately. We initially considered using a line chart for visualization. However, four lines could have made it difficult to distinguish the AES encryptions from each other. By studying the TPM distribution of each encryption algorithm in the diagram, we quickly understood the extent of the overall difference in performance between each AES encryption key length. With ANOVA and Tukey tests we performed a statistical analysis to determine if a significant difference was present, and used the results from this analysis as a basis for answering our first research question.

As we also noticed that the TPM rate varied greatly between the databases, we decided to conduct a second experiment where we investigated the variation through the use of another metric which is explained in more detail in the method section.

To fulfill the ninth and tenth objectives, we configured the stress tests to measure the NOPM produced by each database, and we changed the configuration of the benchmarking tool to now focus on recording NOPM values. Due to time restrictions, we only tested with the AES-192 algorithm, and considering how the difference was so significant between the two databases, we believe that changing the key length would not have severely changed the results in favor of MySQL. To present the results of this experiment we decided to use bar charts to represent the two metrics in a simplistic but understandable manner. When we tested the databases with the NOPM metric we performed another statistical analysis on the results to verify if a significant difference in transaction throughput between the two databases existed, whereby we used this result to answer our second research question.

7.1.3 Security and TPM Trade-offs

As related research studies brought up in this paper lack a thorough and explicit description of how their data analysis process was conducted, we decided to meticulously list the specific algorithms and analysis methods applied in our study. By conducting a statistical analysis through both ANOVA tests and Tukey tests, we could prove any potential difference or indifference that could occur.

Another finding from our analysis was that the unencrypted database had a lower average TPM than many of the encrypted ones. This was observed to be the case in both MariaDB and MySQL. Our findings seem to align with the results observed in the study by Madyatmadja, Hakim, and Sembiring (2021). Where load testing conducted on a relational database encrypted with TDE (Transparent Data Encryption), which is a form of data-at-rest encryption, outperformed the non-TDE encrypted one in terms of transaction rate.

However, in the same study by Madyatmadja, Hakim, and Sembiring (2021), the stress tests suggest that the TDE encrypted database had a higher CPU usage and memory consumption than the non-TDE one. Based on this, we hypothesize that the increased transaction rate during load tests on the TDE encrypted database both in their study and ours could be due to a possible increase in CPU and memory consumption occurring during the load test. Since CPU and memory measurements were omitted in this study, a continuation of this research could further investigate this matter.

This research concluded that the transaction rate was not significant between an encrypted system and a non-encrypted system. However, our results both confirm and contradict this conclusion to some extent. Regarding MariaDB, the results are statistically insignificant, while in MySQL, the results are statistically significant when TPM (transactions per minute) was measured. We argue that data-at-rest implementations with the AES algorithm can produce different results across relational databases. Still, a more thorough investigation that takes hardware metrics such as CPU and memory into account would need to be conducted to explain this phenomenon further.

In the study by S.Tongkaw and A.Tongkaw (2016), MySQL displayed a higher level of performance than MariaDB did. In contrast, our findings suggest the opposite, as our results indicate a significant performance gain in terms of TPM when using MariaDB over MySQL.

The reason for this might have to do with how we employed a later version of MySQL (8.0.29) and that the version used in the study by S.Tongkaw and A.Tongkaw (2016) is an older version (MySQL 5.6) that might include more functionality such as dynamic thread pooling that is now exclusive to the enterprise edition of MySQL.

Throughout our experiment, we collected a broad amount of data used to analyze and test our hypotheses and research questions. During this process, we noticed some additional interesting trends. As can be seen in section 6.4, during each of the load tests performed with different AES key lengths on both MySQL and MariaDB, a noticeable trend seems to occur where the growth in TPM between each set of virtual users does not seem to occur in a linear fashion as the number of virtual users is increased. This drop-off in TPM appears to be less prevalent in the load tests done with MySQL than in the load tests with MariaDB, where the growth seems to level out as the number of virtual users increases. Our current thoughts are that this could be related to how thread pooling is implemented in each database, though we have yet to conclude on any specific reason for this drop-off. As a result, further investigation is needed in order to verify if this is indeed due to the inner workings of each database or other potential hardware and software limitations.

Based on all our findings, we would therefore conclude that there is a difference in transaction rate and that the added security gained through implementing AES-256 to secure data-at-rest outweighs the potential loss of transactional throughput that might occur by doing so. Since the longer key length of the AES-256 provides a higher level of security due to the increased number of potential combinations (2^{256} possible combinations) compared to AES-128 and AES-192 (2^{128} and 2^{192} possible combinations respectively) we would argue there are no severe downsides to choosing AES-256 over any other AES versions.

7.2 Ethical Issues and Validity Threat

7.2.1 Ethics

Everything presented in this report relating to figures, diagrams, or illustrations except figure 2 under the experimentation section is produced by ourselves. The figure that we did not create ourselves originates from the official HammerDB documentation page, and all content on that page is free to distribute as it is an open-source project. The figure borrowed is simply an illustration of the TPROC-C Schema created by the benchmarking tool.

From the DBMS to the benchmarking tool, all software used in this experiment is open-source and created by several contributors. The gathered results and publication of these results do not conflict with any external interest guaranteed by the GPL end-user license. No individuals were involved in this experiment, so there were no ethical aspects regarding people we had to consider. The results found from this research are intended to be seen as a comparative analysis between encryption algorithms in SQL databases implemented for data-at-rest.

The workload schema was generated by the benchmarking tool itself. The only parameter we chose to manipulate was how many "warehouses" the schema should contain and the number of virtual users running the workload. The number of warehouses correlates with the maximum number of virtual users configured to run the workload. The authors of the benchmarking tool give general guidelines on configuring the dataset but highlight that there is no one-size-fits-all solution.

We configured the system based on what we thought was sufficient for the hardware we used for experimentation. However, a more significant number of warehouses does not indicate that the performance will be better. Thus, a decent configuration should be generalizable for anyone who wants to recreate this experiment.

The source code for HammerDB is open source and is available to anyone online from their official website or GitHub repository. Thus, we chose not to disclose it in this research. The diagrams displaying the data were all created in PyCharm using Python with the matplotlib library.

The experiment was conducted on a computer borrowed from the company at which we are writing our thesis. This computer was only used to set up relevant software and perform the tests required for the experiment. The company was not directly involved with the experiment itself or had any special interests in the results gathered from this experiment.

7.2.2 Threats to Validity

Wohlin et al. (2012) give a detailed description of various validity threats, and below are some of the main validity threats identified after evaluating our experiment.

Based on the results we gathered from our experiment, it was observed that the difference in throughput after applying various encryptions in the DBMS was minimal. Given this circumstance, one could not rule out interference from other processes running in the background when we ran the tests. DBMS servers and the benchmarking tool were installed on a computer dedicated solely for this experiment to mitigate this potential validity threat. Thus, the results gathered should not be significantly affected by random noise except for the operating system the computer was running on. The data in the results indicates that the existing outliers are due to virtual user contention and how the DBMS handle transactions rather than interference from other processes.

One threat to internal validity in our experiment is that we had to manually format the throughput data generated by the benchmarking tool when creating the diagrams. We had to do this because the tool itself managed to record empty values before the test began and when the test ended. To mitigate this issue, we recorded each test for four minutes, 1-minute ramp-up time, and 3 minutes of continuous testing. We then deducted the ramp-up time and the last minute from the total testing time, effectively collecting data each second for 2 minutes for each test. This gave us only two zero values to manually remove from each test which we both were present to avoid potential human error.

Another threat to internal validity is the benchmarking tool itself. Several open-source benchmarking tools could apply to our experiment, and each could produce different results based on how they are designed. We chose an open-source benchmarking tool to mitigate this issue, namely HammerDB. This benchmarking tool is used in related studies, is well established in the benchmarking community, and has good documentation.

One threat to external validity in our research is the generalizability of the experiment. This research focused on investigating performance trade-offs in SQL databases that utilize data-at-rest encryption algorithms. To make this research more generalizable in a broader perspective, we could have integrated more SQL databases into our experiment. However, some candidate DBMS we considered, such as Oracle and Microsoft SQL, are commercially exclusive and come with an end-user license agreement that prevents individuals from

disclosing benchmarking results without the company's consent. This predicament led us only to use open-source databases under a GPL (General Public License) license, namely MariaDB and MySQL. The benchmarking tool could also support PostgreSQL, but the DBMS had not integrated inherent data-at-rest support when we were experimenting. Thus, this could only be implemented by using third-party commercial plugins. Despite only using two open-source DBMS, they are popular in the community, and these results could be used to generalize to a broader context of open-source relational databases.

We initially intended to investigate all AES key lengths when we ran the stress tests to compare the transaction throughput between the two databases. The reason for performing stress tests with only one AES key length was mainly due to time restrictions, and as was mentioned in section 6.6, we chose AES-192 as it had the best performance of the load tests done in MySQL. We acknowledge that this was a biased choice, and how we implied based on the TPM results from each load test in MySQL that by choosing AES-192 over AES-128 or AES-256 we would ensure a better and more fair comparison as we assumed that AES-192 would also report higher NOPM values during the stress test for MySQL.

8 Contributions and Future Work

8.1 Contributions

The importance of adequate data security becomes even more vital as more sensitive information is being stored digitally and accessed online daily. This predicament leads both companies and agencies to opt for systems that remain as reliable and secure as possible with a minimum performance loss from a monetary perspective. Therefore, the results gathered from our study could be of help when making these important decisions. With cyber-attacks becoming more and more prevalent in today's society, the deeper insight that this study might contribute to, if ever so slightly, regarding making a better and more informed decision in terms of securing sensitive information is well enough justification for the study to have been conducted in the first place.

Previous related research has primarily focused on data-at-rest encryption in single database management systems. Instead, this study aims to give database technicians and software developers who utilize relational database management systems a better overview of the trade-offs between transactional optimization and database security. This research also displays transparency in how the databases are configured when measuring transactional throughput. How the parameters of the database are tuned is valuable information for people interested in the results and further research about data-at-rest encryption in databases. That is why these aspects have been clearly defined in this research paper.

The results in this research from investigating the AES algorithm focus mainly on the average transactional throughput affected by this encryption algorithm and do not consider different viable encryption algorithms such as Triple DES (data encryption standard) or asymmetric encryption algorithms. However, these findings are relevant because most relational databases mainly support the AES algorithm. These results can, in turn, be used practically in a real-world setting when deciding what level of security is best suited for a project or protection of sensitive data in existing databases.

Previous research by Madyatmadja, Hakim, and Sembiring (2021) shows that transactional throughput increases with data-at-rest encryption implemented in a Microsoft SQL Server. This research can corroborate a part of those findings with entirely different relational databases, namely MySQL and MariaDB. However, in contrast to their study, we investigated the AES algorithm used commonly for implementing TDE (transparent data encryption) and statistically proved the existence of a significant difference in transaction rate between non-encrypted and encrypted databases. This ambiguity of how encryption affects the transaction rate encourages further research in this area.

8.2 Future work

As this study was limited to measuring the average transaction rate per minute in two databases with data-at-rest encryption, specifically MariaDB and MySQL, future studies could aim to incorporate a more extensive set of databases along with additional metrics. Metrics such as the CPU and memory consumption could provide an even deeper insight into the potential cost and trade-offs of implementing data-at-rest encryption with a specific AES-key length.

Further expansion on this study could be done by performing the same experiment with another benchmarking tool's similar or near-identical setup. This could allow for possible validity threats or discrepancies that may have been caused by HammerDB to be identified and avoided. By opting for a different operating system such as Linux, a similar study on AES algorithms could, for instance, be made to include PostgreSQL as this would require plugins currently not supported in a Windows OS environment. By expanding the study to include non-relational SQL databases, future work could be an even more comprehensive comparison to better generalize potential differences in performance between relational and non-relational databases regarding transaction throughput.

In this study, we conducted stress tests to measure the difference in terms of transaction throughput between MariaDB and MySQL by measuring the NOPM per TPM. As this was only measured and compared when AES-192 was applied to MariaDB and MySQL, further research could encompass a larger set of different databases and versions of these databases along with a more considerable amount of different data-at-rest encryptions being tested. Conducting a broader study with the specific metric of NOPM in mind, better visualization of the average transaction throughput across a wide set of databases could aid future developers in making a more informed decision when choosing between RDBMS based on transaction throughput.

References

Advanced Encryption Standard (AES). (2001). [online] U. S. N. I. of Standards and T. (NIST). Available at: https://www.nist.gov/publications/advanced-encryption-standard-aes [Accessed 14 Mar. 2022].

Hammerdb. (2018) "Why both TPM and NOPM Performance Metrics?" *Hammerdb Blog,* December 20, 2018 [Blog]. Available at: https://www.hammerdb.com/blog/uncategorized/why-both-tpm-and-nopm-performance-metrics [Accessed 5 May. 2022].

IBM (No date). *What is a database management system?* [online] www.ibm.com. Available at: https://www.ibm.com/docs/en/zos-basic-skills?topic=zos-what-is-database-management-system [Accessed 1 Mar. 2022].

Jiang, Z.M. and Hassan, A.E. (2015). A Survey on Load Testing of Large-Scale Software Systems. *IEEE Transactions on Software Engineering*, 41(11), pp.1091–1118. doi:10.1109/tse.2015.2445340.

Madyatmadja, E.D., Hakim, A.N. and Sembiring, D.J.M. (2021). Performance testing on Transparent Data Encryption for SQL Server's reliability and efficiency. *Journal of Big Data*, 8(1). doi:10.1186/s40537-021-00520-z.

Meier, A. and Kaufmann, M. (2019). *SQL et NoSQL Databases : Models, languages, consistency options and architectures for Big Data Management*. Wiesbaden Springer Vieweg.

MySQL (No date). *MySQL :: MySQL 8.0 Reference Manual :: 15.13 InnoDB Data-at-Rest Encryption*. [online] dev.mysql.com. Available at: https://dev.mysql.com/doc/refman/8.0/en/innodb-data-encryption.html [Accessed 14 Mar. 2022].

Natarajan, K. and Shaik, V. (2020). Transparent Data Encryption: Comparative Analysis and Performance Evaluation of Oracle Databases. *2020 Fifth International Conference on Research in Computational Intelligence and Communication Networks (ICRCICN)*. doi:10.1109/icrcicn50933.2020.9296168.

Oracle (2021). *What is a relational database?* [online] Oracle.com. Available at: https://www.oracle.com/database/what-is-a-relational-database/ [Accessed 1 Mar. 2022].

Raasveldt, M., Holanda, P., Gubner, T. and Mühleisen, H. (2018). Fair Benchmarking Considered Difficult. *Proceedings of the Workshop on Testing Database Systems*, pp.1–6. doi:10.1145/3209950.3209955.

Samaraweera, G.D. and Chang, M.J. (2019). Security and Privacy Implications on Database Systems in Big Data Era: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 33(1). doi:10.1109/tkde.2019.2929794.

Saraswat, M. and Tripathi, R.C. (2020). Cloud Computing: Comparison and Analysis of Cloud Service Providers-AWs, Microsoft and Google. *2020 9th International Conference System Modeling and Advancement in Research Trends (SMART)*. doi:10.1109/smart50582.2020.9337100.

Tilborg, V. and Jajodia, S. (2011). *Encyclopedia of cryptography and security*. Springer, Boston, MA.

Tongkaw, S. and Tongkaw, A. (2016). A comparison of database performance of MariaDB and MySQL with OLTP workload. *2016 IEEE Conference on Open Systems (ICOS)*. doi:10.1109/icos.2016.7881999.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M.C., Regnell, B. and Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg.

Zaw, T.M., Thant, M. and Bezzateev, S.V. (2019). *Database Security with AES Encryption, Elliptic Curve Encryption and Signature*. [online] IEEE Xplore. Available at: https://ieeexplore.ieee.org/document/8840125 [Accessed 16 Mar. 2022].

Zhu, J., Cheng, K., Liu, J. and Guo, L. (2021). Full encryption. *Proceedings of the VLDB Endowment*, 14(12), pp.2811–2814. doi:10.14778/3476311.3476351.