Performance Testing on Transparent Data Encryption for SQL Server's Reliability and Efficiency

Evaristus Didik Madyatmadja (✉ emadyatmadja@binus.edu)

School of Information Systems, Bina Nusantara University <https://orcid.org/0000-0003-3316-5226>

Aditya Nur Hakim

Bina Nusantara University

David Jumba Malem Sembiring

Universitas Informatika dan Bisnis Indonesia

Research

Keywords: Microsoft SQL Server, Transparent Data Encryption, Performance Testing

Posted Date: April 20th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-409001/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Abstract

Data security is being one of the most crucial aspects to be focused on system development. However, using such a feature to enhance the security of data might affect the system's performance. This study aims to observe how substantial Transparent Data Encryption as a solution for data security on Microsoft SQL Server will affect the database management system's performance. Each of the system performance is conducted with stress and load test. This paper concentrates on the upsides of using Transparent Data Encryption over standard database by finding how significant performance degradation has occurred in terms of Reliability and Efficiency.

1. Introduction

In the age of information explosion, data has already been a part of our daily lives that we are often unaware that we are using one. Data are processed to create information as meaningful and useful content and communicated to a recipient who uses it to make a decision [1]. Therefore, a system called Database Management System (DBMS) is used as a solution for storing and retrieving data [2]. When managing data, the user wants the system application to be providing the needs of their data management at ease. However, the existence of risks in using the system cannot be avoided as our present-day life is vastly driven by data [3]. Various possibilities of data theft can occur when carrying out a particular project such as database server migration, cloning activities on data file storage, natural disasters that may arise, or when trying to retrieve existing data file storage [4]. Hence, the database's security is essential as data is one of the most valuable assets owned and cannot be disclosed by anyone [5]

According to research conducted by Hammouchi [6], from January 2005 to December 2018, the Privacy Rights Clearinghouse (PRC) has recorded more than 9,000 hacking cases containing 12 billion data with many instances. *"Each breach can be caused by an insider that intentionally breaches information (INSD), payment card fraud (CARD), physical loss (PHYS), lost or stolen portable device (PORT), being hacked by someone or infected by malware (HACK), stationary equipment loss (STAT), an unknown method (UNKN) or an unintended disclosure like sending an email to the wrong person (DISC)"* [6]. As much as 13.05% of data loss was caused by data storage being stolen and 2.77% due to missing onsite equipment. This can lead to the possibility of perpetrators to view existing data by taking physical storage places containing confidential and essential files.

As in Fig. 2, data hacking from 2005 to 2009 was experienced by losing physical storage devices and increasing all types of hacks between 2009 and 2013. However, the decline in data hacking decreased as business activists understood how vital data security must be maintained. The PORT and STAT types of hacks remain stable from year to year and have not as many data hacks as other restrictions, but the trend is still there and remains a data security risk.

Being one of the world's largest software vendors, Microsoft designed a DBMS called Microsoft SQL Server. SQL Server is a relational database server that supports the well-known Structured Query Language (SQL) database language [7]. As a publicly known DBMS, SQL Server has the advantage of being streamlined installation, enhanced performance, lower cost of ownership, and better security features. However, it also has several shortcomings in terms of complex performance tuning, no native support of source control, and expensive cost for the enterprise edition. As a security measurement, SQL Server offers one of its features called Transparent Data Encryption (TDE). It encrypts data files and logs in real-time at disk level [8, 9]. This creates a solution where sensitive data can be gated security by using a Database Encryption Key (DEK), which is stored in a database that must be unlocked using a special certificate that was formed when implementing Transparent Data Encryption [10]. This can prevent the possibility of someone using data without having a key and increasing the security of the database system owned by the company by preventing unwanted access, reducing cost for managing user privacy, and provide maximal protection by inhibits the perpetrator to open the data contents of physical files [11].

However, the use of encryption in the database system will impact the performance of a system. As stated by Sharma and Trivedi [12], based on the research that has been done, taking an approach to modeling from several levels for error prevention and safety may require sacrifices based on several attributes such as Efficiency and Reliability. Other research conducted by Wolter and Reinecke showed that the combination of security and performance poses interesting tradeoffs and inspires similar models as the combination of performance and dependability, known as performability [13]. The consequences of security on the probability of having a particular system state require more performance, judging by the increased need for transactions for encryption time, combined performance and security (CPSM), and ongoing transactions.

This research was conducted by initiating performance testing and compare the implemented TDE's SQL Server and non-implemented TDE's SQL Server. We focus on performance value such as Reliability and Efficiency to know how significant performance degradation once a system is implemented by TDE by doing Load Testing, Stress Testing and Backup Testing. Each of test can show how affected the systems are by implementing TDE on database system. Thus, determining on how much percentage of performance would be taken from using TDE and not using TDE. The result from this paper will be achieved using HammerDB as a benchmarking tool and Performance Analysis of Logs (PAL) as an analysis tool. This paper aims to provide knowledge about how Transparent Data Encryption would affect the database system's performance and how substantial the degradation of performance is by gaining a security measurement on the system.

2. Research Methodology

This research will be conducted using Performance Testing, divided into Load Testing, Stress Testing, and Backup Testing—carried out to see the effect of a system's performance when implementing Transparent Data Encryption based on hardware performance. We will compare the result based on two values, which are Reliability and Efficiency. Reliability is assessed based on the number of transactions per minute, and

Efficiency assessed based on CPU and Memory usage in percentage units and the duration for data backup.

For the test itself, we have created a trial model with the same performance to be compared; the difference is only on the implementation of TDE. Here is some detail for the trial model.

The trial model has a similar specification as below:

Table 1
Trial Model Specification

Specification	System 1	System 2
CPU	Intel Core I7-7700HQ CPU @ 2.80Ghz (2 CPUs)	Intel Core I7-7700HQI7-7700HQ CPU @ 2.80Ghz (2 CPUs)
Memory	6 GB	6 GB
OS	Windows Server 2016	Windows Server 2016
DBMS	Microsoft SQL Server 2017	Microsoft SQL Server 2017
Security Measure	With TDE	Without TDE
Memory	6 GB	6 GB
CPU	Intel Core I7-7700HQ CPU @ 2.80Ghz (2 CPUs)	Intel Core I7-7700HQI7-7700HQ CPU @ 2.80Ghz (2 CPUs)
Memory	6 GB	6 GB

We will conduct the Performance Test with HammerDB version 3.3 as a benchmarking tool. It is used as a database load and stress testing by simulating multiple virtual users' workload against the database for transactional and analytic scenarios [14]. HammerDB will create a schema with a structured data type database and became a simulation object for performance testing. Afterward, HammerDB will run a test workload with 4,482,350 total data used as a transaction, documented with a performance counter log, and analyze with PAL ver 2.8.2 as a monitoring tool. PAL is used by analyzing the performance counter log that has been captured and converts it into graphical data. As the database was generated using a model database, we can assume that database is sufficiently provisioned with a resource that allows the performance model to be built [15]. The test will carried out to find that the system is ready for use in a work environment that cannot be disturbed. By knowing system performance from the start, developers can detect early possible design problems and can fix them in detail and precisely [16]

This research will be focused on two values, namely Reliability, and Efficiency. One of the depictions that can be done in technical work on software quality metrics is the Rome Air Development Center (RADC) software quality consumer-oriented attributes. Used to show the classification of requirements from how the system can answer existing problems [17]. The following are functional attributes that have been classified as the purpose of this research

Table 2
Functional Attribute

Value Types	User Problem	Functional Requirement	Test Scenario	Test Parameter
Reliability	How can transactions run smoothly without interruption?	The transaction was stable, and there were no interruptions	Load Testing	Number of Transactions in minute (transaction/min)
Efficiency	How can the system use resources properly?	The system runs with the resources that have been provided (no difference)	Stress Testing,	CPU Usage (%), Memory Usage (%),
			Backup Testing	Backup Duration (ms)

Testing will be carried out by three methods, namely Load Testing, Stress Testing, and Backup Testing. Hardware performance counters are used for capturing the result of the difference between the system used by software engineers for measuring performance and allowing software vendors to enhance their code [18]. As a benchmarking tool, HammerDB is using an automatic task and categorize the benchmark into five randomly selected mixed transactions following the percentage as follows:

- 1 New-Order: Receiving new orders from customers (45%),
2. Payment: Updating accounts customer to record payment (43%),
3. Delivery: Giving orders asynchronously (4%),
4. Order-Status: Receiving customer status based on the latest order (4%), and
5. Stock-Level: Providing status of warehouse storage (4%).

The following is an explanation of the three methods used to perform Performance Testing.

Load Testing is done to look for differences in a system's performance with a significant load, aiming to look for performance and functional problems under load pressure [19]. Load Testing can be measured through several criteria, one of which is Transaction Rate. Thus, Load Testing can help testers to perform assessments on the Reliability type. For Load Testing that will be performed, the examiner will simulate transactions on database file in system 1 using TDE and system 2 without TDE. The number of transactions given is 100,000 transactions, and then it will be seen the number of transactions that are running using the Transaction Counter query as a reference in determining the Reliability value.

Stress Testing has the meaning to put enormous pressure on a system for quite a long time. The reason for doing Stress Testing is to find out the behavior of the application when facing pressure that is greater than expected to face daily activities when there is a large enough pressure [16]. Running a load that is large enough for an extended period makes it easy to identify problems that may occur, such as memory, CPU, storage space, or other needs. We will simulate transactions on the database used on system 1

using TDE and system 2 without TDE. Transactions will run for 30 minutes with four virtual users. Simultaneously, the performance monitor will perform data retrieval to assess the performance of the two systems. Then the results will be processed and analyzed using monitoring tools, namely PAL. By using PAL, we can see system performance in terms of CPU utilization and memory consumption as a reference in assessing the Efficiency value. Thus, stress testing can help the examiner to determine the Efficiency value.

Backup Testing aims to see how the backup process can be affected by the performance of a system after implemented security measurement, Transparent Data Encryption. The reason for the Backup Testing process is to find out how a system can restore data when there is a spontaneous activity, such as data loss or others. However, a backup test can also be done to find out how long this activity will be carried out by a system because the action to perform a backup can be affected by the performance of the existing database system [20]. We will conduct the test by doing a database backup process using compression and not using compression on system 1 using TDE and system 2 without TDE. The database size that will be backed up is 1.232MB or about 1.2GB. We will then record the time for the backup process by displaying the time before and after the backup in milliseconds. We use SQL Server Management Tools (SSMS) to perform backup queries and gain time to perform backup activities. Thus, Backup Testing can help the examiner to determine the Efficiency value

3. Result And Discussion

Performance Testing is carried out in stages starting from system 1, followed by system 2, which begins with Load Testing, then continues with Stress Testing, and ends with Backup Testing. The parameters used in Performance Testing are as follows: Load Testing looks at the number of transactions per minute. Stress Testing looks at the percentage of CPU usage when performing tests, the percentage of Memory usage when performing tests, and Backup Testing looks at the time per minute when backing up the database.

Load Testing is done by running the HammerDB application as a benchmarking tool and observing how many transactions that run when completing 100,000 transactions that are conducted three times for more detailed and definite reference. The results obtained are transactions per minute for each system. It is used as a reference for analyzing the average output of each test. The purpose of this test is to see whether the TDE system and the TDE system have an effect seen from the transactional rate. These are the result of Load Testing:

Figure 4 above shows that system 1 and system 2 transactions have a not-so-significant difference when observed from the number of transactions executed. The transaction runs for 22 minutes with 2 minutes as a ramp-up time or as preparation time for the benchmark tool to prepare its test transaction execution. System 1 gets an average transaction rate of 3,555 per minute, while system 2 receives an average transaction rate of 3,623 per minute. The following is a table of conclusions from the Load Testing.

Table 3
Comparison of Load Testing Values

	TDE (system 1)	Non-TDE (system 2)	Transaction Gap	Transaction Gap (%)
Average transactions per minute	3555	3623	-68	2%
Minimal transactions per minute	3129	3007	122	-4%
Maximal transactions per minute	3906	4219	-312	7%

From the table above, it can be concluded that the result from the average transaction that runs every minute is 68 transactions per minute or 2% of the total transactions per minute. From the results, it can be seen that the usage of Transparent Data Encryption will affect the non-implemented Transparent Data Encryption's system up to 7%.

Stress testing is done by running a performance monitor to retrieve the required data with the Data Collector Set and the HammerDB application as benchmarking tools. The data retrieved includes information about the average read and write speed of the logical disk, memory consumption in terms of available memory, and CPU usage. The result obtained is a file in the form (.blg), which contains information about the system's performance for 30 minutes, running with four virtual user transaction processes against the database. The purpose of this test is to see whether the Transparent Database Encryption system and non-Transparent Data Encryption system has a difference in the form of an effect on system performance seen from the use of hardware. These are the result of Stress Testing.

Figure 5 shows that system 1 and system 2 hardware usage have a not-so-significant difference when observed from the percentage of CPU usage and memory consumption. Based on the test conducted, the result shown that the average CPU usage on system 1 is 29%, average CPU usage on system 2 is 21%, average memory consumption on system 1 is 35%, and average memory consumption for system 2 is 34%. The two results above indicate that the use of TDE has an insignificant effect on system performance. The following is a comparison table of the two data obtained after testing

Table 4
Comparison of Stress Testing Values

		System 1 (TDE)	System 2 (Non-TDE)	Performance Gap	Performance gap (%)
CPU Usage	Average	29%	21%	8%	
	Minimal	12%	14%	-2%	
	Maximal	42%	37%	5%	
Memory Usage (MB)	Average	2161	2086	-75	2%
	Minimal	2236	2180	-56	2%
	Maximal	2125	1978	-147	4%

From the table above, it can be concluded that the amount of difference that exists when viewed from the average CPU usage that runs for 30 minutes is 8%, and the average memory usage is 2%. From the results, it can be seen that the use of Transparent Data Encryption will affect the non-implemented Transparent Data Encryption's CPU Usage up to 8% and Memory Usage up to 4%

Backup Testing was conducted by running a backup query to perform the database backup process. The backup process is done by saving the time before doing the backup activity and after doing the backup activity and comparing it in milliseconds. The following are the results of the tests.

Figure 6 shows that the two tests with compression and without compression have a difference between system 1 and system 2. Systems using TDE seem to require more time than systems that do not use TDE. It might be happening because the backup process is encrypted by the Database Encryption Key so that it takes a longer time for the security of the backup file provided. The following are the details of the tests.

Table 5
Comparison of Backup Testing Values

Backup Type	Compression		Non-Compression	
System	System 1 (TDE)	System 2 (Non-TDE)	System 1 (TDE)	System 2 (Non-TDE)
Duration (Ms)	26121	23540	28344	23183
Gap Duration (%)	90%		82%	
Based on the table above, the use of TDE in system 1 increases the backup process duration compared to system 2 that does not use TDE. The total duration increases by about 10–18% depending on whether or not compression is used when performing backups.				

The conclusion of Performance Testing is Transparent Data Encryption has a little degradation in terms of system performance by a bit of margin. The differences can be seen from the percentage of CPU

usage, memory, and duration when doing backup activities. The conclusion from the performance testing that the author has done is as follows:

Table 6
Conclusion of Performance Testing Trials

Testing Type	Criteria	Gap Difference	Percentage Difference
Load Testing	Transaction Rate	68 transaction/min	2%
Stress Testing	CPU Usage	8%	
	Memory Usage	75 MB	2%
Backup Testing	Duration Time	3871 ms	15%

The difference in performance varies between 2–15% depending on what activity is being carried out, and this test is carried out in a position where the buffer pool does not have any data because the new database is formed, so the results can be the worst-case scenario. When conducting experiments on systems that have been running for a long time, the performance results can improve compared to this test's results.

4. Conclusion

This paper has shown that Transparent Data Encryption develop a performance degradation for database system. The conclusions that can be drawn from the trial are as follows:

Transparent Data Encryption uses approximately 2%-15% of system databases' performance on each value focused on, namely Reliability and Efficiency. For Reliability value, it appears that Transparent Data Encryption will be degrading the system performance up to 7% of the transaction rate per minute. For Efficiency, it seems that Transparent Data Encryption will be degrading the system performance up to 15% of CPU, Memory, and Backup Duration. However, the benefit of using Transparent Data Encryption for security measurement is considered useful and adds another layer of security for the system's data. We concluded that Transparent Data Encryption is practical to use regardless of the degradation performance shown on the system because the advantages of Transparent Data Encryption can be a consideration based on the feature that it has.

Declarations

Ethics Approval

No need ethics approval.

Consent to Participate

No need consent to participate

Availability of data and Material

Data is attached in the form of rar data (png and .xlsx)

Competing interests

We don't have a financial and non-financial competing interests must be declared in this section.

Funding

This work is supported by Research and Technology Transfer Office, Bina Nusantara University

Authors' Contribution

Evaristus Didik Madyatmadja : Data Collection, writing the manuscript, analysis and interpretation data

Aditya Nur Hakim : Programmer

David Jumpa Malem Sembiring : Data Collection, writing the manuscript

Acknowledgement

Thanks to Bina Nusantara University who has funding support.

References

1. S. B. Gupta e A. Mittal, Introduction to Database Management System, Delhi: University Science Press, 2017.
2. H. Jagadish, A. Chapman, A. Elkiss, M. Jayapandian, Y. Li, A. Nandi e C. Yu, "Making database systems usable," in: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, pp. 13-24, 2007.
3. A. P. A. G. Deshmukh, "International Journal of Advanced Computer Science and Applications," Transparent Data Encryption- Solution for Security of Database, pp. 25-27, 2011.
4. D. Cherry, Securing SQL Server: Protecting your Database from Attackers, Syngress, 2011.
5. U. T. Mattson, "A Practical Implementation Of Transparent Encryption and Separation Of Duties in Enterprise Databases: protection against external and internal attacks on databases," Seventh IEEE International Conference on E-Commerce Technology (CEC'05), pp. 559-565, 2004.
6. H. Hammouchi, O. Cherqi, G. Mezzour, M. Ghogho e M. E. Koutbi, "International Symposium on Machine Learning and Big Data Analytics for Cybersecurity and Privacy," Digging Deeper into Data Breaches: An Exploratory Data Analysis of Hacking Breaches Over Time, pp. 1004-1009, 2019.
7. S. Varga, D. Cherry e J. D'Antoni, Introducing Microsoft SQL Server 2016: Mission-Critical Application, Deeper Insights, Hyperscale Cloud, Microsoft Press, 2016.
8. M. Coles e R. Landrum, Expert SQL Server 2008 Encryption, Springer Natur, 2011.

9. S. Mukherjee, "Popular SQL Server Database Encryption Choices," SSRG International Journal of Computer Science and Engineering (SSRG-IJCSE), vol. 66, n° 1, pp. 1-6, 2018.
10. K. Guyer, V. To, G. Milener e M. Ray, "Transparent Data Encryption (TDE)," 09 05 2019. [Online]. Available: <https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-ver15>.
11. N. Alain, A. KIBE e C. W. K., "International Journal of Scientific Engineering and Technology," Use of Enhanced Transparent Data Encryption to Protect Database Against Exposure of Backup Data, pp. 477-481, 2016.
12. V. S. Sharma e K. S. Trivedi, "Quantifying software performance, reliability and security: An architecture-based approach," The Journal of Systems and Software 80, pp. 193-509, 2006.
13. K. Wolter e P. Reinecke, "Performance and Security Tradeoff," Formal Methods for Quantitative Aspects of Programming Languages Volume 6154, pp. 135-167, 2010.
14. S. Shaw, "About: HammerDB," November 2019. [Online]. Available: <https://hammerdb.com/about.html>. [Acesso em 6 May 2020].
15. R. Oshman e W. J. Knottenbelt, "Database system performance evaluation models: A survey," Performance Evaluation, vol. 69, n° 10, pp. 471-493, 2012.
16. S. Wilson e S. Lin, "Techniques for Testing Performance/Scalability and Stress-Testing ADF Applications," Oracle White Paper, California, 2011.
17. L. Chung, B. A. Nixon, E. Yu e J. Mylopoulos, Non-Functional Requirements in Software Engineering, New York: Springer, 2000.
18. U. Leif, G. Andy e V. Ingrid, "Exploiting Hardware Performance Counters," 5th Workshop on Fault Diagnosis and Tolerance in Cryptography, pp. 59-67, 2008.
19. N. Thakur e K. L. Bansal, "Load Testing on Web Application using Automated Testing Tool: Load Complete," International Journal of Innovative Research in Computer and Communication Engineering, pp. 9305-9315, 2015.
20. P. D. Guise, Enterprise systems backup and recovery : a corporate insurance policy, Florida: CRC Press, 2009.
21. S. D. Negara, "The Indonesian Economy under Jokowi's Second Term," ISEAS Yusof Ishak Institute, 2019.
22. s. IT, "DB-Engines Ranking," 1 December 2019. [Online]. Available: <https://db-engines.com/en/ranking>.
23. R. Abdukalykov, I. Hussain, M. Kassab e O. Ormandjieva, "Quantifying the Impact of Different Non-Functional Requirements and Problem Domains on Software Effort Estimation," IEEE 2011 9th International Conference on Software Engineering Research, Management and Applications (SERA), pp. 159-165, 2011.
24. S. Kansal e M. Mittal, "Performance evaluation of various symmetric encryption algorithms," International Conference on Parallel, Distributed and Grid Computing., pp. 105-109, 2014.

25. H. Boral, "A Methodology for Database System Performance Evaluation," ACM SIGMOD Record, vol. 14, n° 2, pp. 176-185, 1984.

26. J. G. R. Ramakrishnan, Database Management Systems, Boston: McGraw-Hill, 2003.

27. T. Herath e H. R. Rao, "Encouraging information security behaviors in organizations : Role of penalties, pressures and perceived effectiveness," Decis. Support Syst, vol. 47, n° 2, pp. 154-165, 2009.

28. B. L. Sirisha, S. S. Kumar e B. C. Mohan, "Steganography based information security with high embedding capacity," 2015 National Conference on Recent Advances in Electronics & Computer Engineering (RAECE), pp. 17-21, 2015.

29. C. Huffman, "clinhuffman / PAL," 1 October 2018. [Online]. Available: <https://github.com/clinhuffman/PAL>.

Figures

Figure 1

Distribution of Hack Types from 2005 to 2018

Data Breach by Type

Figure 2

Hack Type Trends from 2005 to 2018

Figure 3

Research Object

Figure 4

Comparison of Average Load Testing Result

Figure 5

Comparison of Average Stress Testing Results

Figure 6
Comparison of Average Backup Testing Results

Figure 7

Conclusion of Performance Testing

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [simulation.rar](#)