Madyatmadja *et al. J Big Data* (2021) 8:134
https://doi.org/10.1186/s40537-021-00520-z

Journal of Big Data

RESEARCH Open Access

Performance testing on Transparent Data Encryption for SQL Server's reliability and efficiency

Evaristus Didik Madyatmadja[1*], Aditya Nur Hakim[1] and David Jumpa Malem Sembiring[2]

*Correspondence: emadyatmadja@binus.edu
[1] Information Systems Department, School of Information Systems, Bina Nusantara University, Jakarta, Indonesia
Full list of author information is available at the end of the article

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.

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

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*

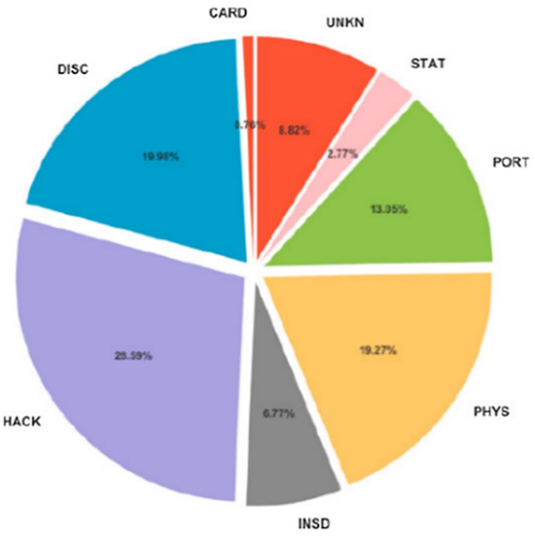

Fig. 1 Distribution of hack types from 2005 to 2018

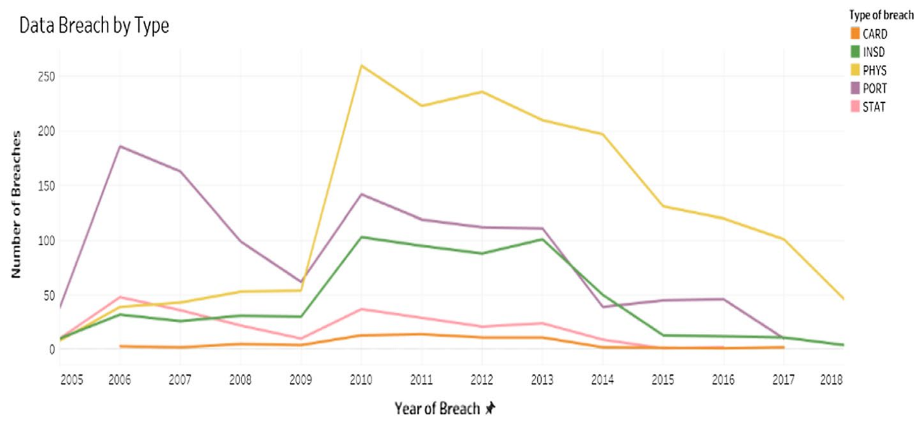

Fig. 2 Hack type trends from 2005 to 2018

(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. 1, distribution of hack types from 2005 to 2018.

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].

Related work

Big data encryption

Privacy and security in the context of big data are critical issues. In the case of complicated applications, the big data security paradigm is not recommended, hence it is disabled by default. However, in its absence, data can always be easily corrupted [12].

Big data may contain sensitive information about persons, privacy is a crucial consideration. To address the privacy issue, data can be de-identified by deleting traits that would allow an individual to be identified. This is a technique that, when done effectively, works both while data is controlled and when it is released [13].

Encryption is a powerful method for ensuring data security. The essence of data encryption is to use algorithms to convert the original plaintext file or data into an unreadable string of code known as ciphertext. Even if someone intercepts the distorted code, he or she cannot utilize it to obtain the original message. This efficiently preserves the secrecy of the data and prevents tampering with the data Users with access can decrypt the file using the matching private key, then update or modify the ciphertext. There are two types of encryption: symmetric encryption and asymmetric encryption. To encrypt and decrypt data, symmetric encryption employs a secret key [14].

Encryption also is the most widely discussed approach, and it can protect data secrecy and integrity. Because not all encryption methods are created equal, cloud service providers and users must employ the most recent encryption techniques (Homomorphic encryption, AES or DES) and longer keys, which strain processor capabilities [15].

Transparent data encryption concept

Shmueli and Vaisenberg investigate five traditional database encryption architectures and compare them. The writers illustrate that existing design can offer a high degree of security, have a considerable effect on performance and impose large changes on the application layer or can be transparent to the application layer and provide great performance [16].

However, the use of encryption in the database system will impact the performance of a system. As stated by Sharma and Trivedi [17], 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 [18]. 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.

Transparent data encryption may help, however solutions for TDE supplied with important database systems only ensure a data-only system, and are seemingly unnecessary if the adversary can access the computer physically, which poses a likely concern when hosting in the cloud. This work provides an alternative approach to TDE, taking into consideration cloud-specific hazards, extends encryption to cover data in use and partially information in motion and is able to run huge SQL sub-sets including heavy relationship operations, complex attribute and transaction operations [16].

Transparent encryption technology allows data to be encrypted throughout the process without altering user habits. It's an encryption algorithm that is also stressed in encryption as "transparent." The window system now potentially have a useful application with transparent encryption technologies. The hook software intercepts the opening function of the file when the user opens the file. The file is copied to the hidden directory folder, decrypted data and provided to reader to obtain a clear copy before the file information is sent to the reader. The hook application also can intercept the closing process if the user shuts the file, encrypts the file above, before saving to the storage device and then transfers it to the original folder. This completes the transparent process of the complete document encoding and decryption [19].

A study done by [20] focuses on Transparent Data Encryption, a technique that is used to tackle data security issues. Transparent encryption implies that databases are encrypted on a hard drive and on any backup medium. Today there are many security dangers and compliance problems in the global corporate world need security solutions that are transparent by design to defend against data theft and fraud. Transparent Data Encoding provides a transparent, standard-based security system that secures network, disk and media data. By transparently encrypting data it is straightforward and efficient to safeguard the stored data. High security levels for columns, tables and tables that are database files saved on hard drives or floppy drives or CDs, and other protection information.

Transparent Data Encryption (TDE) offers transparent, standard-based security for network, disk, and backup data protection. By transparent encryption of data, it is easy and efficient protection of stored data. TDE is able to encode and decrypt data and log files in real-time. The encryption employs a Database Encryption Key (DEK), which is saved for recovery in the database boot. The DEK is a symmetric key encrypted with an EKM module protected certificates in the server's master data base or with an asymmetric key. TDE secures 'rest' data, which means data and log files. It enables many rules, regulation and guidelines made in different industries to be complied with [21].

In fact, TDE works effectively, if the backup of your database to be protected. You need a master key, a certificate to restore if you are implementing TDE in the source server and wish to restorate your database to another server. Think about opening your bank locker. One key is to implement an extra layer of protection with you and the other key is the prohibition specialist. The Always Encrypted (AE) allows transparent encryption of client apps from the database. This AE function is enhanced by TDE by the addition of an enclosure layer in the memory and transit of sensitive data as well as in rest. In fact, the Always Encrypted Driver encrypts and decrypts the application. Any potential leakage to database administration can therefore be managed by the information owner by keeping the decryption keys, in order to prevent administrators from accessing sensitive data. In contrast, the database administrator uses the master key and certificates to access the TDE encryption keys [22].

Thus, determining on how much percentage of performance would be taken from using TDE and not using TDE. SQL interface encryption implemented is: (1) the sensitive table is renamed; (2) the sensitive table is encrypted, (3) the encryption trigger is defined; (4) the decryption view has been defined. In theory, the application layer should be transparent in this architecture. In practice it is not, however, as: (1) some actions cannot be executed on the view and must be reprogrammed to utilize the renaming table (for example, insert, update or truncate). (2) No questioning of range is supported. The aforementioned cache design is to be implemented in MySQL, and only by modifying two strategic areas in the InnoDB storage engine: the cache location (added decryption) and the cache value location (add encryption). Similar to the aforementioned cache architecture, the storeroom architecture implementation requires only two strategic points in an InnoDB storage device to be changed: the location where a site reads from the disk (i.e. all cells in this page will be decrypted) and the location of the site on the disk (i.e., encrypt all cells in this page) [16].

Reliability and efficiency

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.

Challenged with greater hurdles than demands for results, reliability and availability. For example, failure rates for software systems are exceedingly difficult to assess unless software testing is improved [23].

Reliability, availability models, and recovery and maintenance times are as well as models that are used to drive the models, such as failure rates, recovery success rates. The state of SW systems can be very vast to limit the application of analytical models of availability. Simulation models with substantial approximations therefore become the only means for testing availability and reliability [23].

Performance testing

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.

The performance of a machine learning-based approach, particularly modern machine learning, is well recognized to depend on several exercise models. This variety can be expressed by number of subjects in a given scenario, such as our case. In this part, the impact on recognition performance of the number of training subjects is examined [24].

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.

Figure 3 shows the research object in this study. Table 1 shows the trial model has a similar specification

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

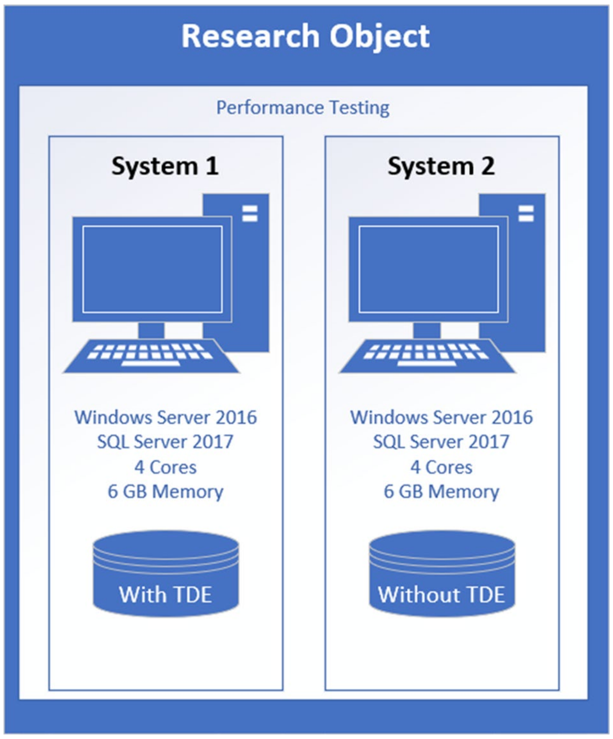

Fig. 3 Research object

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

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, Backup Testing	CPU Usage (%), Memory Usage (%), Backup Duration (ms)

users' workload against the database for transactional and analytic scenarios [25]. 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 [26]. 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 [26].

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 [27]. Table 2 shows functional attributes that have been classified as the purpose of this research.

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 [28]. 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 [29]. 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 [27]. 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 min 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 [30]. 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 1.232 MB or about 1.2 GB. 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

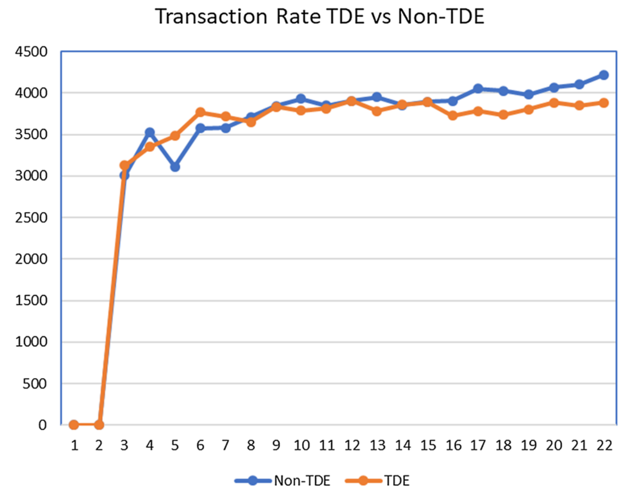

Fig. 4 Comparison of average load testing result

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

perform backup activities. Thus, Backup Testing can help the examiner to determine the Efficiency value.

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 min with 2 min 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 3623 per minute. Table 3 shows conclusions from the load testing.

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 min, 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%. CPU Usage is a term used to describe how much the processor is working. Memory Consumption is the amount of memory a particular program utilizes throughout its execution.

The two results above indicate that the use of TDE has an insignificant effect on system performance. As in Table 4, comparison table of the two data obtained after testing.

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 min is 8%, and the average

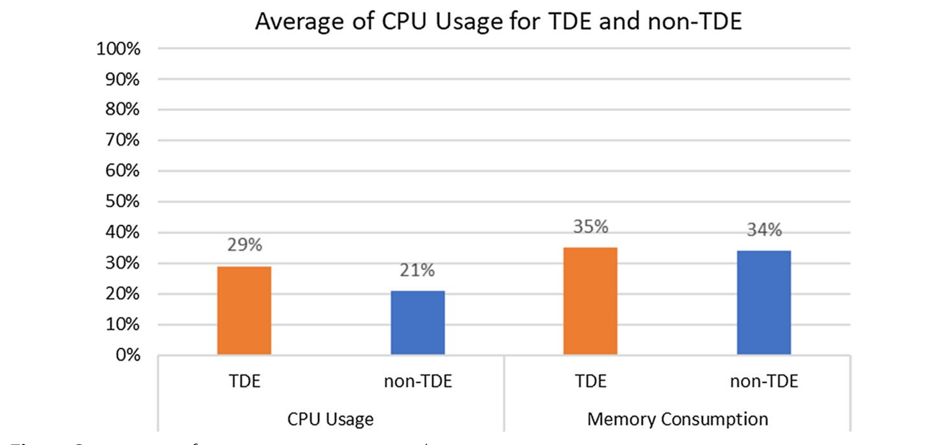

Fig. 5 Comparison of average stress testing results

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

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. Compression is the process of reduction in bit number required for representing a data. 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

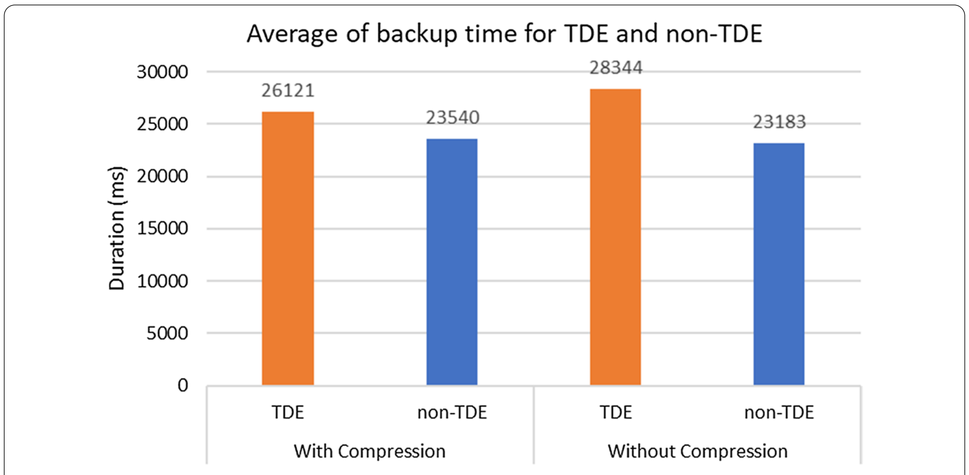

Fig. 6 Comparison of average backup testing results

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)	26,121	23,540	28,344	23,183
Gap duration (%)	90%		82%	

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%

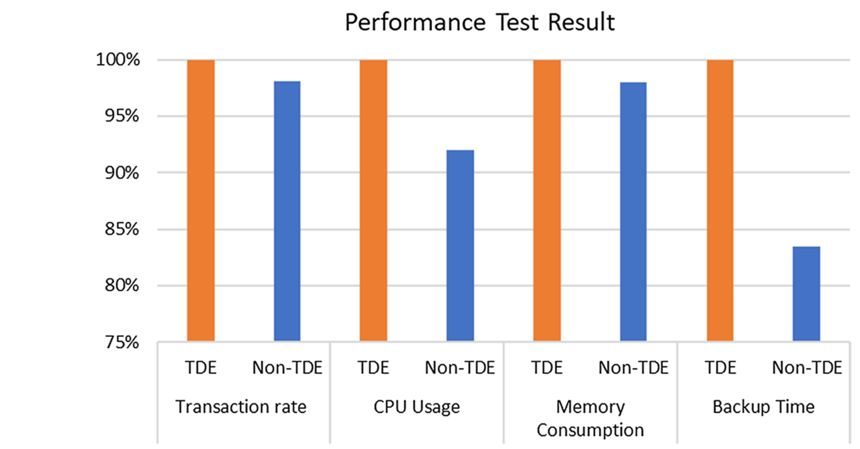

Fig. 7 Conclusion of performance testing

time for the security of the backup file provided. Table 5 shows comparison of backup testing values.

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 shows in Table 6.

The difference in performance varies between 2 and 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.

Conclusion

This paper has shown that Transparent Data Encryption develop a performance degradation for database system. Figure 7 shows conclusion of performance testing.

Transparent Data Encryption uses approximately 2–15% of system databases' performance on each value focused on, namely Reliability and Efficiency. Transaction rate is number of writes, backup time is time of the process of backing up the operational state, architecture and stored data of database software. 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. For the future research can try to implement using SQL Server Big Data Clusters to make it better in implementing big data.

Acknowledgements
Thanks to Bina Nusantara University who has funding support.

Authors' contributions
EDM: Data collection, writing the manuscript, analysis and interpretation data. ANH: Programmer. DJMS: Data collection, writing the manuscript. All authors read and approved the final manuscript.

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

Availability of data and materials
Data is attached in the form of rar data.

Declarations

Ethics approval and consent to participate
No need ethics approval and consent to participate.

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

Author details
[1]Information Systems Department, School of Information Systems, Bina Nusantara University, Jakarta, Indonesia. [2]Teknik Informatika, Institut Teknologi dan Bisnis Indonesia, Medan, Indonesia.

Received: 13 April 2021 Accepted: 27 September 2021
Published online: 18 October 2021

References

1. Gupta B, Mittal A. Introduction to database management system. Delhi: University Science Press; 2017.
2. Jagadish H, Chapman A, Elkiss A, Jayapandian M, Li Y, Nandi A, Yu C. Making database systems usable. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. 2007; p. 13–24.
3. Deshmukh APAG. Transparent data encryption—solution for security of database. Int J Adv Comput Sci Appl. 2011, p. 25–27.
4. Cherry D. Securing SQL Server: protecting your database from attackers. Burlington: Syngress; 2011.
5. Mattson UT, 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). 2004; p. 559–565.
6. Hammouchi H, Cherqi O, Mezzour G, Ghogho M, Koutbi ME. Digging deeper into data breaches: an exploratory data analysis of hacking breaches over time. International Symposium on Machine Learning and Big Data Analytics for Cybersecurity and Privacy. 2019; p. 1004–1009.
7. Varga S, Cherry D, D'Antoni J. Introducing Microsoft SQL Server 2016: mission-critical application, deeper insights, hyperscale cloud, microsoft press. London: Pearson; 2016.
8. Coles M, Landrum R. Expert SQL Server 2008 Encryption, 2011; Springer Natur.
9. Mukherjee S. Popular SQL server database encryption choices. SSRG Int J Comput Sci Eng. 2018;66(1):1–6.
10. Guyer K, To V, Milener G, Ray M, Transparent Data Encryption (TDE)," 09 05 2019. https://docs.microsoft.com/en-us/sql/relational-databases/security/encryption/transparent-data-encryption?view=sql-server-ver15.
11. Alain N, Ann KIBE, Cheruiyot WK. Use of enhanced transparent data encryption to protect database against exposure of backup data. Int J Sci Eng Technol. 2016;5:477–81.

12. Jain P, Gyanchandani M, Khare N. Big data privacy: a technological perspective and review. J Big Data. 2016. https://doi.org/10.1186/s40537-016-0059-y.
13. Joshi JBD. Security and privacy for big data: a systematic literature review. Institute of Electrical and Electronics Engineers, and IEEE Computer Society, Proceedings, 2 IEEE International Conference on Big Data. 2015
14. Nelson B, Olovsson T. Security and privacy for big data: a systematic literature review. IEEE Computer Society and Institute of Electrical and Electronics Engineers, ICISS 2015 : International Conference on Information Science and Security : 2015 in Seoul, Kor 2015.
15. Yang P, Xiong N, Ren J. Data security and privacy protection for cloud storage: a survey. IEEE Access. 2020;8:131723–40. https://doi.org/10.1109/ACCESS.2020.3009876.
16. Shmueli E, Vaisenberg R, Gudes E, Elovici Y, Sidorov V, Ng WK. Transparent data encryption for data-in-use and data-at-rest in a cloud-based database-as-a-Service Solution. Proc. - 2015 IEEE World Congr. Serv. Serv. 2015;44(2): 33–50, https://doi.org/10.1109/SERVICES.2015.40.
17. Sharma VS, Trivedi KS. Quantifying software performance, reliability and security: an architecture-based approach. J Syst Softw. 2006;80:193–509.
18. Wolter K, Reinecke P. Formal methods for quantitative aspects of programming languages. Perform Secur Tradeoff. 2010;6154:135–67.
19. Pang P, Aourra K, Xue Y, Li Y, Zhang Q. A transparent encryption scheme of video data for android devices. Proc. - 2017 IEEE Int Conf Comput Sci Eng IEEE/IFIP Int Conf Embed Ubiquitous Comput CSE EUC. 2017; 1: 817–822.https://doi.org/10.1109/CSE-EUC.2017.163
20. Anwar D, Riyazuddin D. Transparent data encryption—solution for security of database contents. Int J Adv Comput Sci Appl. 2011;2(3):25–8. https://doi.org/10.14569/ijacsa.2011.020305.
21. Husain R, Security of database contents using transparent data encryption in microsoft sql server enterprise edition. International J Adv Comput, 2012; p. 97–108.
22. Mukherjee S. Popular SQL server database encryption choices. Int J Comput Trends Technol. 2019;66(1):14–9. https://doi.org/10.14445/22312803/ijctt-v66p103.
23. Malkawi MI. The art of software systems development: Reliability. Availab vol. Maintaina, no. Performance (RAMP). 2013; 1–17.
24. Uddin MZ, et al. The OU-ISIR large population Gait Database with real-life carried object and its performance evaluation. IPSJ Trans Comput Vis Appl. 2018. https://doi.org/10.1186/s41074-018-0041-z.
25. Shaw S. About: HammerDB. November 2019. https://hammerdb.com/about.html. Acessed 6 May 2020.
26. Oshman R, Knottenbelt WJ. Database system performance evaluation models: a survey. Assoc Comput Mach. 2012;69(10):471–93.
27. Wilson S, Lin S. Techniques for testing performance/scalability and stress-testing ADF applications. California: Oracle White Paper; 2011.
28. Chung L, Nixon BA, Yu E, Mylopoulos J. Non-functional requirements in software engineering. New York: Springer; 2000.
29. Leif U, Andy G, Ingrid V. Exploiting hardware performance counters. 5th Workshop on Fault Diagnosis and Tolerance in Cryptography. 2008; 59–67.
30. Thakur N, Bansal KL. Load testing on web application using automated testing tool: load complete. Int J Innov Res Comput Commun Eng. 2015;9305–9315

Publisher's Note