IBIMA Publishing
Journal of Information Assurance & Cyber security
https://ibimapublishing.com/articles/JIACS/2021/614511/
Vol. 2021 (2021), Article ID 614511, 9 pages, ISSN: 2165-9923
DOI: 10.5171/2021.614511

Research Article

Database Encryption for Balance Between Performance and Security

André Gomes[1], Carla Santos[2], Cristina Wanzeller[3] and Pedro Martins[4]

[1,2]Polytechnique of Viseu, Viseu, Portugal

[3,4]Research Centre in Digital Services, Polytechnique of Viseu, Viseu, Portugal

Correspondence should be adressed to: Pedro Martins; pedromom@estgv.ipv.pt

Received date: 24 August 2020; Accepted date:20 January 2021; Published date: 9 April 2021

Academic Editor: Roberto Paiano

Abstract

In an increasingly digital world, information security is a very recurring theme and a growing concern for companies. This involves the protection of data and confidential or non-confidential information of a company, which transit between all its sectors and between the organization and its stakeholders. We can say that the pillars of information security are integrity, confidentiality, and availability. But this constant concern brings an increase in complexity, reflected in the performance of the systems. To study the impact of security on the performance of systems, operations were tested on databases with and without encryption and the average times and occupied space were analyzed. There was a substantial impact on the reading and writing performance of the database when it uses encryption, this impact being directly proportional to the level of security used.

Keywords: Security, Database, Cryptography, Performance.

Introduction

To make a performance analysis for the use of encryption in a relational database, an attempt was made to find a model that would strike a good balance between performance and security. The encryption of databases can be done in different ways: the data can be encrypted at the table level; row; or column level.

For this performance study, PostgreSQL was used with the native module pgcrypto. This database was chosen because it has good results in performance tests concerning other

Cite this Article as: André Gomes, Carla Santos, Cristina Wanzeller and Pedro Martins (2021), "Database Encryption for Balance Between Performance and Security", Journal of Information Assurance & Cyber security, Vol. 2021 (2021), Article ID 614511, DOI: 10.5171/2021.614511

databases (Ohlsson & Persson 2019). Also, because of its flexibility to choose the encryption method, analyze issues such as security level, study the performance of insertions and queries. Note that also the total storage space was taken into account.

We started by using PGP encryption, which is known to be secure on email systems (Kar et al. 2019) and other messaging platforms (Ilmonen 2020), but in our preliminary tests proved to be inefficient for large volumes of data, so it was discarded. The best solution found was the use of raw encryption with AES (Bhange & Mathur 2019), which, being simpler, proved to be more efficient. This type of encryption is used in several current scenarios, as is the example of smart cities (Farahat et al. 2019). This decision naturally leads to a decrease in security. Even so, there was a decrease in performance compared to tests without encryption.

The data read results, queering, show that, with encryption, the times increase as there is a need for decryption. As expected, with the same queries, the use of primary and foreign keys allowed the reduction of time, but with encryption, the same did not happen for all tested queries.
Another analysis that is important to consider is the difference in the space occupied by the data, which was found to increase in scenarios with encryption and keys.

This document is organized into four sections. In Section 2, we perform a study of different forms of database encryption. In Section 3, is explained the experimental scenario. Section 4 is where the case study results are presented. Finally, Section 5 concludes the work, and we present possible future work guidelines.

Related work

In this section, some works carried out in this area are analyzed. These help to identify alternatives and improvements to the current state-of-the-art.

Asymmetric Keys

Since data security is an issue of increasing importance, in this case, study, the use of cryptography algorithms is tested. These algorithms have become a constant concern for improvement, which has led to an increase in complexity involving higher execution times that are reflected in a decrease in application performance. Thus, we make a comparison of the execution times of three algorithms using asymmetric keys, depending on the size of the keys, used in different scenarios and database domains, Rivest-Shamir-Adleman (RSA) (Sharma et al. 2019) (Bhange & Mathur 2019), ElGamal (Li 2020) and Elliptic Curve Integrated Encryption Scheme (ECIES) (Saravanan & Venkatachalam 2019).

The results in (Boicea et al. 2017) show that the encryption time increases with the number of encrypted characters, but with special attention to RSA where the encryption time varies very little with the increase in the number of characters. It was also analyzed that the algorithms have encryption times of the same order of magnitude, considering the length of the encryption key. The decryption time is dozens of times less than the encryption time for all algorithms and increases with the size of the key, again the RSA with shorter times is highlighted.

AES to IoT

In the IoT scope, security is also a criterion to take into account, after being received, the encrypted data from the IoT sensors are saved in text format. However, suppose the integrity, validity and security of the data are to be guaranteed. In that case, these encryption and decryption operations cause overhead in the data query time, which, in turn, result in additional operational delays and increased consumption of data power in wireless IoT devices.

In (Singh et al. 2017), the encryption performance of a database that contains relational tables and tables with BJSON data fields. Since the commonly used benchmarks do not include tests of encrypted column databases, the authors created a set of Python scripts to test PostgreSQL AES-128bit encryption in CBC (Hameed et al. 2019) mode, currently used for data transmission wireless IoT de- vices. Analyzing this case study and the present, both use symmetric key encryption algorithms and AES (Bhange & Mathur 2019) encryption. The works differ in the type of data, in which relational and non-relational BJSON data fields are used.

As for the results in (Kokkonis et al. 2019) for data with BJSON, they showed that the average insertion time is the same for encrypted and unencrypted. How- ever, for queries, an 8% increase in processing time has been noted for encrypted documents compared to plain text documents.

FNR encryption

FNR encryption storage preserves the length and format of the fields. For this, an encrypted database was used in CryptDB using the FNR encryption scheme. This scheme consists of the classification of the data type, which is encrypted in a text with the same size that is again converted into data with the format of the original. To this end, scenarios are tested to explore the feasibility of Format Preserving Encryption (FPE) (Pérez-Resa et al. 2020). Using SQL Aware Encryption, an encryption scheme was defined for each predefined set of SQL operations, so all data items are encrypted so that any operation is possible without the need for decryption.

In comparison to the present scenario, and in addition to all the application and technologies being different, the encryption itself is also different. This allows the preservation of the size and format of the fields in a way that does not require decryption in a set of SQL operations, something that is considered inevitable in most forms of encryption, including those used in the context of this document.

In the experimental results (Chandrashekar et al. 2015), improvements in storage were measured, with a decrease in performance. Because, in terms of storage, they showed a decrease of approximately 50% concerning AES. Although this value always depends on the data of the chosen application, it was found that the size of the data with FNR was always equal to the size in normal text, in contrast to the AES which was always much larger. The performance, of insertion operations, of the FNR when compared to the AES-128 was about seven times higher since the FNR scheme uses several encryption steps internally. Thus, it is concluded that encryption using FNR reduces the size of the database because the size of the data remains. Still, it leads to a decrease in performance in data insertions because it requires several steps to reach the final encrypted data.

Experimental Setup

For the experimental scenario, 10GB of data were generated with the TPC-H. The interaction with the database was carried out through Python using the SQL Alchemy module, so for insertions the Python script that reads each row of data from each table and converts it into an insertion command with 4kB of bulk.

For encryption scenarios, this command changes: for each field, the encryption function is added. In this way, the loading was tested, with and without encryption, to measure the difference in performance and space occupied by the data.

As for the consultations, five queries of different complexities of the TPC-H were chosen (Q1, Q2, Q3, Q4, Q5), where for each scenario they are executed five times, and the average is calculated discarding the maximum and the minimum. These are also executed by a Python script where, in the case of

encryption, it was necessary to add decryption functions to the data so that operations could be performed on them.

In the experimental scenario, the module pgcrypto (Odongo & Bukenya 2019) was used, which provides cryptographic functions for PostgreSQL (Juba & Volkov 2019). This module provides several forms of encryption. Initially, the PGP encryption model was tested using symmetric keys. In this model, the password provided is a hash calculated from a String2Key (S2K) algorithm (Shakya & Karna 2019). The data, to be encrypted must undergo a manipulation that includes compression, conversion to UTF-8 and/or conversion of line terminations. Then the data is prefixed with a block of random bytes, a SHA1 hash of the random prefix and the data is attached and placed in a data pack. This model was discarded as soon as data was entered, and times increased exponentially, which became impractical times for large amounts of data. Therefore, raw encryption functions were used, using the AES algorithm, which only performs a cypher on the data and does not have any advanced PGP encryption feature. This model directly uses the user's key as the cypher key and does not provide any data integrity checks. Although it does not deal with text, having to be converted to bytes, this model proved to be more efficient for insertion.

Experimental Results

In this section, the results are presented in the form of charts and their analysis is carried out, so it will be divided into two sub-sections, insertions and queries. In the first, the encryption scenario using PGP, the AES encryption scenario and the same with the use of keys is be presented. For queries, AES encryption is used, with and without keys.

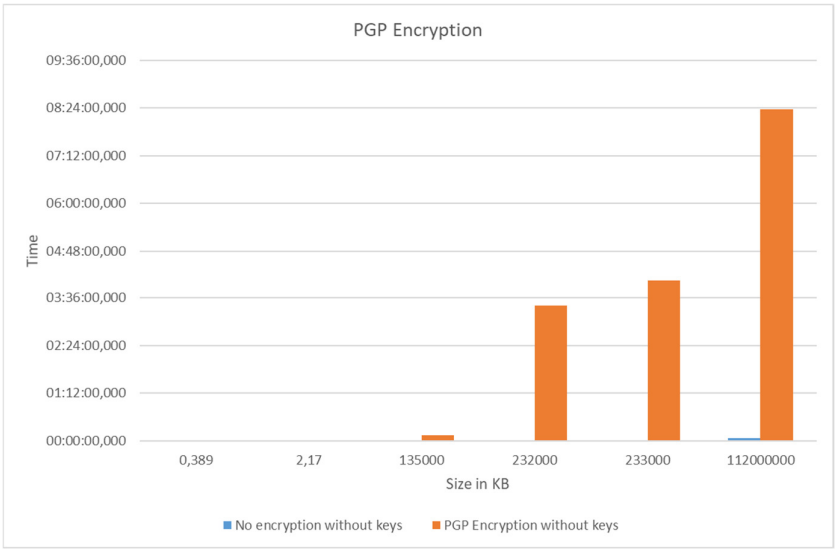

Fig. 1: Chart encryption PGP

Insertion Performance

The results of the insertions in scenarios with and without encryption are presented to evaluate the performance differences. In 4.1.1 the PGP encryption method is tested. In 4.1.2 and 4.1.3, the raw encryption method is tested with and without AES, and with the use of keys, respectively.

PGP Encryption

Through the chart 1 the insertion times with the PGP model are shown using symmetric keys in comparison to the

insertion without encryption.

As can be seen in figure 1, a high increase began to be noted with only 13.5 MB of data which worsened dramatically by 1.12 GB. These times can be explained by the numerous steps, described above, for encrypting a single row of data. Thus, it is concluded that, despite being good encryption that allows the verification of integrity, it is impractical to use this encryption for large volumes of data.

AES Encryption

Since PGP encryption proved to be impractical for this volume of data, crude encryption functions were used with the use of AES. The chart 2 shows the performance of this encryption compared to the same scenario without encryption.

This encryption model (Figure 2) proved to be more efficient than the previous one, since comparing both for 1.12 GB of data, this one lasts around 11 minutes, while in the previous one lasted 8 hours. Compared to the scenario without encryption, the times remained close to 1.12 GB of data, noticing a dispersion that grew with the increase of the data volume. Concerning total times, without encryption, it took 1:18:34 h, while for this encryption it took 05:23:27 h, about five times longer.

In conclusion, this encryption mode is less secure, as it has no integrity check. On the other hand, it proved to be more efficient than the previous one, already compared to the scenario without encryption, with the increase of the data volume, the performance is degraded.

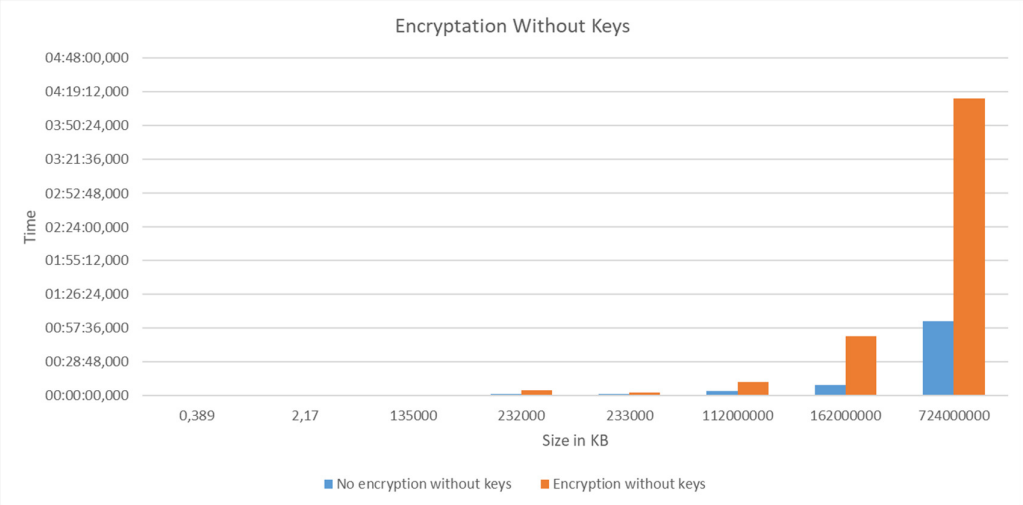

Fig 2. Chart encryption AES

AES encryption with keys

The previous scenario was repeated, including the use of primary and foreign keys, and the times are illustrated in the chart 3.

About the scenario without encryption with keys, the conclusions are the same as in the previous one, but the increase in time concerning the increase in volume proved to be even more significant, reaching 25:13:21 h for 7.24GB. Thus, the scenario without

encryption with keys took a total of 03:33:15 h, and with encryption and keys, it was 27:34:11 h, an increase of around nine times.

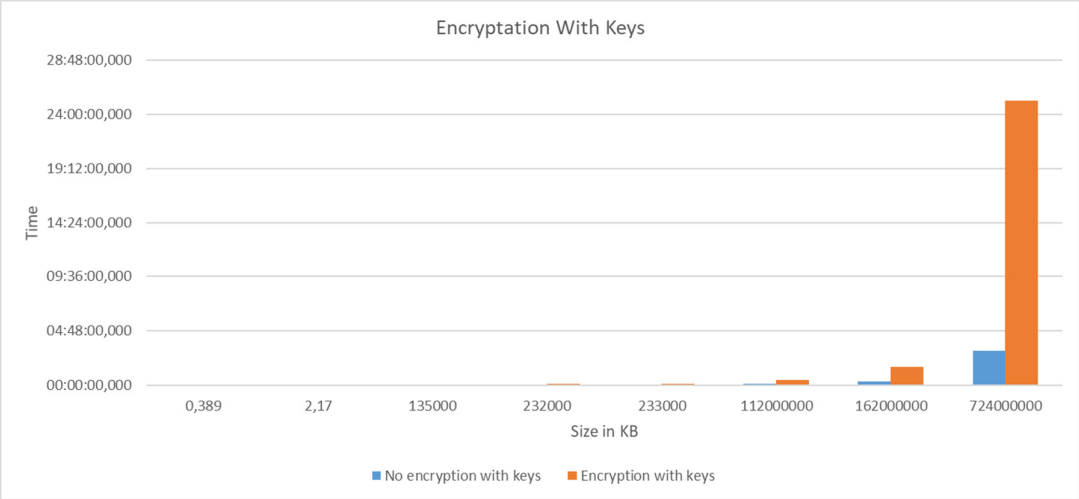

Fig. 3: Chart encryption AES with keys

Performance in Queries

This section is analyzed the performance of data queries with and without encryption.

In chart 4, the average execution time without keys is shown for each query. With encryption, all queries have increased execution times, being more visible in some more than others due to the amount of data required, explained by the amount of data that is accessed to perform them, as they need decryption to perform operations, or to process the join operations, as well as the amount of data to perform the final data projection.

The same query scenario was performed for encryption with primary and foreign keys, and the average execution times can be seen in the chart 5. Compared to the same scenario without encryption, the times were higher, as expected. Still, while in scenarios without encryption, the execution times decreased with the use of keys, in scenarios with encryption, the same did not always happen. There were three queries in which the times increased with the use of the keys.

Occupied storage space by the database

In Table 1, the occupied storage space is studied. When comparing the space occupied by the databases with and without keys or indexes, there is a natural increase in the use of encryption, as this, regardless of the type of data, converts into a bit chain with sizes that are mostly larger than the original. The increase was about 3.8 times. For databases with the use of keys, the same difference was noted, with an increase of approximately four times.

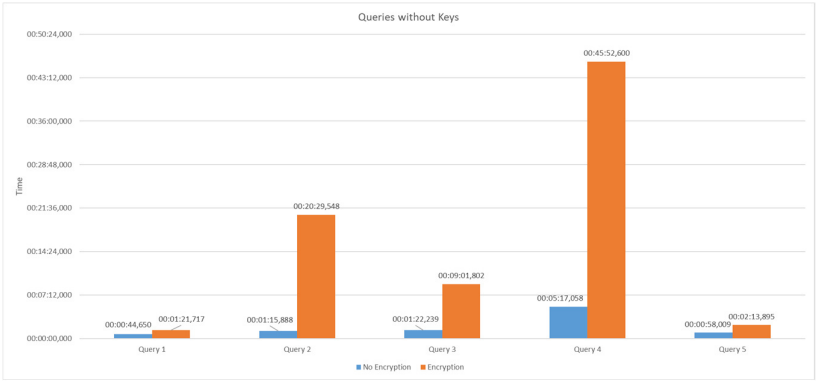

Fig 4. Chart queries without keys

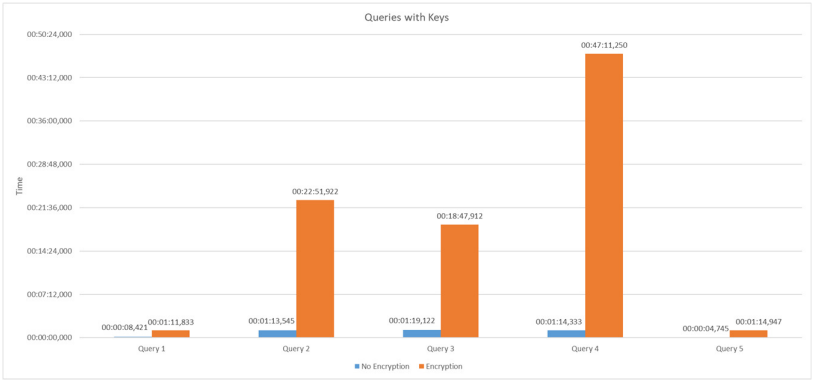

Fig. 5: Chart queries with key

Table 1: Database Storage

DB Size (in GB)	No encryption without keys	Encryption without keys	No encryption with keys	Encryption with keys
	13	49	14	58

Conclusions and Future Work

With the security level being the most important criterion, an algorithm with fewer security breaches may involve a longer encryption and decryption time, depending on the key size. This leads to significant reductions in performance, additional consumption of resources on the server and increased storage space for encrypted data. Another important aspect is key management because if decryption can only be done with the corresponding encryption key, there is a risk that the administration will lose data due to failure, and the entire database is compromised.

In a search for encryption that would strike a better balance between security and performance, the solution went through the use of raw encryption with the use of AES. Despite being less secure encryption, as it does not have data integrity validation, it proved to be more efficient, but still with performance losses quite accentuated as the data increases.

It is concluded that the loss of performance in data encryption is

inevitable. Still, depending on the requirements of the scenario to be implemented, the best balance between the desired security and performance should be sought. In the case presented, this happened when there was a decision to change between PGP and AES encryption.

For future work, it is important to analyze this encryption mode for larger volumes of data, to continue the evaluation of the identified performance degradation with the increase in data. Additionally, it would be interesting to compare the impact of encryption in different databases.

Acknowledgements

"This work is funded by National Funds through the FCT Foundation for Science and Technology, I.P., within the scope of the project Ref UIDB/05583/2020. Furthermore, we would like to thank the Research Centre in Digital Services (CISeD), the Polytechnic of Viseu for their support."

References

- Bhange, M. A. & Mathur, H. (2019), 'Comparative analysis of several cryptography algorithm with its effectiveness towards the security and its performance', Journal of the Gujarat Research Society 21(6), 42–46.
- Boicea, A., Radulescu, F., Truica, C.-O. & Costea, C. (2017), Database encryption using asymmetric keys: a case study, in '2017 21st International Conference on Control Systems and Computer Science (CSCS)', IEEE, pp. 317–323.
- Chandrashekar, P., Dara, S. & Muralidhara, V. (2015), Efficient format preserving encrypted databases, in '2015 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)', IEEE, pp. 1–4.
- Farahat, I., Tolba, A., Elhoseny, M. & Eladrosy, W. (2019), Data security and challenges in smart cities, in 'Security in Smart Cities: Models, Applications, and Challenges', Springer, pp. 117–142.
- Hameed, M. E., Ibrahim, M. M., Manap, N. A. & Attiah, M. L. (2019), 'Comparative study of several operation modes of aes algorithm for encryption ecg biomedical signal.', International Journal of Electrical & Computer Engineering (2088-8708) 9.
- Ilmonen, M. (2020), Platform-Agnostic End-to-End Encryption for Modern Instant Messaging Platforms, PhD thesis, University of Aberdeen.
- Juba, S. & Volkov, A. (2019), Learning PostgreSQL 11: a beginner's guide to building high-performance PostgreSQL database solutions, Packt Publishing Ltd.
- Kar, J., Naik, K. & Abdelkader, T. (2019), 'An efficient and lightweight deniably authenticated encryption scheme for e-mail security', IEEE Access 7, 184207– 184220.
- Kokkonis, G., Asiminidis, C., Georgiadis, I., Syndoukas, D. & Kontogiannis, S. (2019), Measurements on encrypted and non-encrypted iot datasets stored in relational and jsonb fields, in '2019 4th South-East Europe Design Automation, Computer Engineering, Computer Networks and Social Media Conference (SEEDA-CECNSM)', IEEE, pp. 1–6.
- Li, L. (2020), An electronic voting scheme based on elgamal homomorphic encryption for privacy protection, in 'Journal of Physics: Conference Series', Vol. 1544, IOP Publishing, p. 012036.
- Odongo, F. O. & Bukenya, D. (2019), Ucudir dspace technical documentation, Technical report.
- Ohlsson, A. & Persson, M. (2019), 'A comparison in performance between a selection of databases', LU-CS-EX 2019-04 .
- Pérez-Resa, A., Garcia-Bosque, M., Sánchez-Azqueta, C. & Celma, S. (2020), 'A new method for format preserving encryption in high-data rate communications', IEEE Access 8,

21003–21016.

- Saravanan, S. & Venkatachalam, V. (2019), 'Privacy preserving of intermediate dataset using hybridisation of oppositional gravitational search algorithm and elliptic curve cryptography', International Journal of Business Information Systems 31(2), 265–281.
- Shakya, A. & Karna, N. (2019), Enhancing md5 hash algorithm using symmetric key encryption, in 'Proceedings of the 3rd International Conference on Cryptography, Security and Privacy', pp. 18–22.
- Sharma, K., Agrawal, A., Pandey, D., Khan, R. & Dinkar, S. K. (2019), 'Rsa based encryption approach for preserving confidentiality of big data', Journal of King Saud University-Computer and Information Sciences.
- Singh, S., Sharma, P. K., Moon, S. Y. & Park, J. H. (2017), 'Advanced lightweight encryption algorithms for iot devices: survey, challenges and solutions', Journal of Ambient Intelligence and Humanized Computing pp. 1–18.