Applied Artificial Intelligence
An International Journal

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uaai20

Enhancing the Time Performance of Encrypting and Decrypting Large Tabular Data

Nguyen Thon Da & Ho Trung Thanh

To cite this article: Nguyen Thon Da & Ho Trung Thanh (2021) Enhancing the Time Performance of Encrypting and Decrypting Large Tabular Data, Applied Artificial Intelligence, 35:15, 1746-1754, DOI: 10.1080/08839514.2021.1991661

To link to this article: https://doi.org/10.1080/08839514.2021.1991661

APPLIED ARTIFICIAL INTELLIGENCE
2021, VOL. 35, NO. 15, 1746–1754
https://doi.org/10.1080/08839514.2021.1991661

Taylor & Francis
Taylor & Francis Group

Enhancing the Time Performance of Encrypting and Decrypting Large Tabular Data

Nguyen Thon Da [a,b] and Ho Trung Thanh [a,b]

[a]University of Economics and Law, Ho Chi Minh City, Vietnam; [b]Vietnam National University, Ho Chi Minh City, Vietnam

ABSTRACT

In the field of data analysis, encrypting and decrypting datasets must keep the information confidential. Currently, encrypting sizable tabular datasets is time-consuming. This study proposes a solution that helps encrypt extensive tabular data in lesser time than that required in conventional methods while preserving data analysis information. We use the feature by which a large dataset can be split into many files in hdf5 format and choose an encrypted algorithm to solve it. The study contributed to information technology knowledge management. We introduce a solution for small-scale companies to encrypt their extensive tabular data economically. The experimental results on three large datasets showed that our solution has a processing time between 1.2–5 times faster than the conventional processing time under some specific situations. The research results assist companies or individuals with a limited financial capacity to deploy data security and analysis at a low cost with time efficiency. The study opens several research opportunities in protecting large datasets and analyzing them in less time.

ARTICLE HISTORY
Received 9 February 2021
Revised 5 October 2021
Accepted 6 October 2021

Introduction

Currently, processing and analyzing big data are compelling issues. Numerous massive datasets consist of up to billions of rows. Many of them contain critical and private information and need to be strictly protected. However, encrypting data's timely execution is one of the severe issues during data processing and transmission (Gai et al. 2016). Several small and limited financial capacity startups face the processing and analysis of extensive data. If they hire compute processing services on a computer network or cloud services, they must pay a prohibitive fee. (Mazrekaj, Shabani, and Sejdiu 2016) Compared and discussed several models and pricing schemes from different cloud computing vendors. Their study indicated that the customers must spend vast amounts of money on hiring such services. Thus, a cost-effective solution is required for small-scale companies or startups. The proposed solution is to apply the veax technique that operates on Python and Jupiter

CONTACT Nguyen Thon Da dant@uel.edu.vn University of Economics and Law, Ho Chi Minh City, Vietnam

Notebook environments. It performs specific stages of splitting data into hdf5 format and then encrypting, decrypting, and opening data. This study secures data with time efficiency while ensuring no loss of information in preparation for the following data analysis phase. The proposed solution can significantly help individuals or companies with limited capital. Besides, it can perform extensive data analysis without spending much money to invest in information technology infrastructure.

Literature Review

Recently, several researchers have proposed various methods to solve the problems regarding data security. (Hazra et al. 2015) suggested a pseudorandom permutation method. They proposed an attached algorithm that produces a block cypher using the mentioned procedure.

(Jueneman et al. 2016) proposed a method for encrypting and sealing a plaintext file by hashing it.

(Gai et al. 2016) proposed a method to address the privacy issues of big data for application in cloud computing. They introduced an algorithm known as D2ES (Dynamic Data Encryption Strategy) to maximize the efficiency of privacy protection. The experimental evaluations showed that the proposed algorithm had an adaptive and superior performance.

(Vishwanath, Peruri, and He 2016) considered data security a critical factor and implemented the AES (Advanced Encryption Standard) algorithm in fog computing. The experimental results showed that all datasets could be processed within a fraction of time, irrespective of their size and type.

(Panda 2016) evaluated both symmetric (AES, DES, Blowfish) and asymmetric (RSA) cryptographic algorithms by taking different files (binary, text, and image files). The experimental results showed that AES performs better than other algorithms for throughput and encryption-decryption time.

(Erofeev and Pawar 2016) introduced for automatically encrypting files. The method can detect access to a first file and determine whether the entry contains convenient access. Besides, it can determine whether the file metadata satisfies the set of encryption rules.

(Maitri and Verma 2016) proposed a security mechanism that requires less time compared to other algorithms, such as AES and Blowfish.

(Aono et al. 2017) presented a privacy-preserving deep learning system in which several learning participants perform neural network-based deep learning.

(Harba 2017) proposed three types of encryption to exploit the advantages of each one in developing a high-security system. They concluded that the overall encryption run is fast and straightforward with low computational costs and provides high system security.

(Rachmawati et al. 2018) They have proposed a hybrid encryption algorithm to address security issues of lightweight data on cloud storage services. They improved the AES algorithm by merging AES and RSA algorithms.

(Yang et al. 2018) introduced a File Remotely keyed Encryption and Data Protection scheme. The technique consists of three-party interactions: a mobile terminal, private clouds, and public clouds.

(Liu et al. 2018) proposed a multi-user verifiable searchable symmetric encryption scheme that achieves all the desirable features of a verifiable searchable symmetric encryption and allows multiple users to perform searches. The experimental results indicated that their scheme shows high performance on some real datasets.

(Sajay, Babu, and Vijayalakshmi 2019) have proposed a hybrid algorithm to improve RSA encryption and decryption speed. Their algorithm also solves the fundamental management problem in the AES algorithm. Besides, it can provide security for users' lightweight data stored in the cloud environment.

(Zou et al. 2020) presented a hybrid encryption algorithm by combining AES and RSA algorithms to solve file encryption efficiency and security problems. Their experimental results showed that RSA and AES's hybrid encryption algorithm has the advantages of algorithm performance and security.

(Mahmoud, Hegazy, and Khafagy 2018) proposed an approach to improve the performance of encrypting/decrypting files using AES and OTP (One Time Passcodes) algorithms integrated on Hadoop. Encryption/decryption in the previous studies used the AES algorithm; the encrypted file size increased by 50% from the original file size. They concluded that their proposed approach improved this ratio because the encrypted file size increased by 20% from the original file size.

Some drawbacks of the existing methods are (1) Encryption, and decryption in the cloud environment cost IT infrastructure-leasing outside; (2) Equipping a multi-computer system for distributed processing, grid computing is also an expensive solution.

Most researchers admitted that the AES algorithm is essential for encryption and decryption. In the scope of this study, we use the AES algorithm for encrypting and decrypting datasets.

Proposed Model

For the tabular data encryption and decryption model for data analysis, encryption will require much time processing big data. The typical model has four main phases: Transforming, encrypting (using AES), decrypting (using AES), and opening. However, this method also faces a significant

problem because it takes a long time to encrypt and decrypt data in hdf5. To solve this, several researchers converted data to hdf5 format to retrieve and analyze data more efficiently.

In Phase 1 (transforming), the large dataset is split into a unique hdf5 file. In Phases 2 and 3 (encrypting and decrypting), the large file is encrypted and decrypted. The sizable hdf5 file is opened in the final phase to prepare for the following process, i.e., the data analysis process.

The proposed model divides tabular data into multiple smaller sized hdf5 files. Then, the smallest size hdf5 file is encoded and decoded.

The proposed model has four main phases: Splitting, encrypting, decrypting, and opening.

In Phase 1 (splitting), the large dataset is split into N parts. Each part is a small hdf5 file. In Phases 2 and 3 (encrypting and decrypting), the smallest file with the smallest size is encrypted and decrypted. In the final phase, by applying the values technology, all hdf5 files are collected to prepare for the following process, i.e., the data analysis process.

Experimental Evaluation

The experiments have conducted on a laptop (Lenovo 541 w) with an Intel i7 processor (8 cores), 32 GB of available RAM, and an SSD drive connected with SATA (Serial Advanced Technology Architecture) in format. The program implements a 64-bit version of Ubuntu 16.04.5 LTS (Xenial Xerus) and the Python programming environment with Anaconda and Jupiter Notebook.

Datasets

Green-trip data (2017–2020) dataset (Number of rows: 27,817,909. Size: 2.69 GB): The green taxi trip records include fields capturing pick-up and drop-off dates/times, pick-up and drop-off locations, trip distances, itemized fares, rate types, payment types, and driver-reported passenger counts. The data used in the attached datasets were collected and provided to the NYC Taxi and Limousine Commission.[1]

ATC-tracking (2012) dataset (Number of rows: 47,398,149. Size: 2.92 GB): The ATC pedestrian-tracking dataset is provided as CSV files; one file for each day (file names are in the format atc-YYYYMMDD.csv).[2]

Power-networks (2015) dataset (Number of rows: 167,932,474. Size: 10.7 GB): UK Power Networks is a distribution network operator for electricity covering South East England, the East of England and London.[3]

Experimental Results

We have conducted some experiments to compare the performance of the existing model and the proposed solution.

(1) After converting the ATC-tracking (2012) into one file in hdf5 format, we obtain the time execution as follows:

Transforming: 62.24916 (seconds)
Encrypting: 89.59078 (seconds)
Decrypting: 39.39382 (seconds)
Opening: 0.03440 (seconds)

Let T_ACT be the total execution time for the four phases (transformation, encrypting, decrypting, and opening) of converting the ATC-tracking (2012) dataset in CSV format into a unique hdf5 file. Thus, T_ATC = 191.26816 (seconds).

Assuming T1_ATC (CHUNK_SIZE = 10_000_000), T2_ATC (CHUNK_SIZE = 1000_000), and T3_ATC (CHUNK_SIZE = 100_000), respectively, to be the total execution time for the four phases (splitting, encrypting, decrypting, and opening) of converting the ATC-tracking (2012) dataset in CSV format into a smaller set of hdf5 files. Thus, T1_ATC, T2_ATC, and T3_ATC are 61.30306, 39.08126, and 100.64416 seconds, respectively (Table 1).

(1) After converting the green-trip data (2017–2020) into one file in hdf5 format, we obtain the time execution as follows:

Transforming: 93.39398 (seconds)
Encrypting: 71.80500 (seconds)
Decrypting: 105.71326 (seconds)
Opening: 0.06859 (seconds)

Table 1. Time execution on ATC-tracking (2012) dataset.

CHUNK_SIZE	PHASES	EXECUTION TIME (seconds)
10_000_000 (5 small files)	Splitting	50.00743
	Encrypting	6.03345
	Decrypting	5.06743
	Opening	0.19475
1000_000 (48 small files)	Splitting	37.41191
	Encrypting	0.456137
	Decrypting	0.39903
	Opening	0.81419
100_000 (474 small files)	Splitting	91.03008
	Encrypting	0.22482
	Decrypting	0.21703
	Opening	9.17223

Table 2. Time execution on green-trip data (2017–2020) dataset.

CHUNK_SIZE	PHASES	EXECUTION TIME (seconds)
10_000_000 (3 small files)	Splitting	119.23783
	Encrypting	17.47493
	Decrypting	15.06482
	Opening	0.58454
1000_000 (28 small files)	Splitting	125.13555
	Encrypting	2.00561
	Decrypting	1.67345
	Opening	4.57668
100_000 (279 small files)	Splitting	187.14510
	Encrypting	0.16527
	Decrypting	0.16704
	Opening	35.74142

Assuming T_TRIP to be the total execution time for the four phases (transformation, encrypting, decrypting, and opening) of converting the ATC-tracking (2012) dataset in CSV format into a unique hdf5 file. Thus, T_TRIP = 270.98083 (seconds).

Assuming T1_TRIP (CHUNK_SIZE = 10_000_000), T2_ TRIP (CHUNK_SIZE = 1000_000), and T3_ TRIP (CHUNK_SIZE = 100_000), respectively, to be the total execution time for the four phases (splitting, encrypting, decrypting, and opening) of converting the green-trip data (2017–2020) dataset in CSV format into a smaller set of hdf5 files. Thus, T1_TRIP, T2_TRIP, and T3_TRIP are 152.36212, 133.39129, and 223.21883 seconds, respectively (Table 2).

(1) After converting the power-networks (2015) dataset into one file in hdf5 format, we obtain the time execution as follows:

Transforming: 802.47295 (seconds)
Encrypting: 416.06674 (seconds)
Decrypting: 388.74097 (seconds)
Opening: 0.31182 (seconds)

Assuming T_POW to be the total execution time for the four phases (transformation, encrypting, decrypting, and opening) of converting the power-networks (2015) dataset in CSV format into a unique hdf5 file. Thus, T_POW = 1607.59248 (seconds).

Assuming T1_POW (CHUNK_SIZE = 10_000_000), T2_ POW (CHUNK_SIZE = 1000_000), and T3_POW (CHUNK_SIZE = 100_000), respectively, to be the total execution time for the four phases (splitting, encrypting, decrypting, and opening) of converting the power-networks (2015) dataset in CSV format into a smaller set of hdf5 files. Thus, T1_POW, T2_POW, and T3_TRIP are 469.64353, 517.81159, and 1007.14603 seconds, respectively (Table 3).

Table 3. Time execution on power-networks (2015) dataset.

CHUNK_SIZE	PHASES	EXECUTION TIME (seconds)
10_000_000 (17 small files)	Splitting	446.26111
	Encrypting	11.21199
	Decrypting	9.32258
	Opening	2.84785
1000_000 (168 small files)	Splitting	502.80226
	Encrypting	1.17626
	Decrypting	1.00076
	Opening	12.83231
100_000 (1680 small files)	Splitting	851.45472
	Encrypting	0.16392
	Decorating	0.15133
	Opening	155.37606

Experimental Evaluation

- With the green-trip data (2017–2020) dataset, when CHUNK_SIZE is 10_000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 1.78 times faster than that required to convert the dataset into a single hdf5 file. When CHUNK_SIZE is 1000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 2.03 times faster than that required to convert the dataset into a single hdf5 file. When CHUNK_SIZE is 100_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 1.21 times faster than that required to convert the dataset into a single hdf5 file.
- With power-networks (2015) dataset, when CHUNK_SIZE is 10_000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 3.42 times faster than that required to convert the dataset into a single hdf5 file. When CHUNK_SIZE is 1000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 3.11 times faster than that required to convert the dataset into a single hdf5 file. When CHUNK_SIZE is 100_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, data presentation) is 1.6 times faster than that required to convert the dataset into a single hdf5 file.
- With ATC-tracking (2012) dataset, when CHUNK_SIZE is 10_000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 3.12 times faster than that required to convert the dataset into a single hdf5 file. When CHUNK_SIZE is 1000_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, and data presentation) is 4.89 times faster than that required to convert the dataset into a single

hdf5 file. When CHUNK_SIZE is 100_000, the total time (to split into multiple smaller files hdf5, encrypt, decrypt files smallest hdf5, data presentation) is 1.9 times faster than that required to convert the dataset into a single hdf5 file.

The test results showed that when CHUNK_SIZE is 1000_000 (in all three cases handled more than three datasets), the total processing time is best (effective in terms of time) because the proposed model always processes the time 2–5 times faster than conventional models.

Conclusion and Future Work

This study proposed a solution to improve the encryption and decryption of sizable tabular datasets on three real tabular datasets. To improve the performance in terms of time, we split the original file into many parts, encrypted and decrypted the smallest file, then used the vaex technology to open them quickly. The experimental results showed that the proposed approach performs better than the existing methods based on execution time. The proposed solution can help small-scale companies to secure their data before transmission via the internet without spending much time and money.

In the future study, we plan to improve this solution to address similar issues regarding large multimedia datasets by developing the existing encryption algorithms.

Notes

1. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
2. https://irc.atr.jp/crest2010_HRI/ATC_dataset/.
3. https://data.london.gov.uk/publisher/timeline/uk-power-networks.

Disclosure Statement

No potential conflict of interest was reported by the author(s).

Funding

This work was supported by the University of Economics and Law, VNUHCM.

ORCID

Nguyen Thon Da http://orcid.org/0000-0002-2660-5011
Ho Trung Thanh http://orcid.org/0000-0002-9033-3735

References

Aono, Y., T. Hayashi, L. Wang, and S. Moriai. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13 (5):1333–45.

Erofeev, A., and R. S. Pawar. 2016. Automatic file encryption: Google patents.

Gai, K., M. Qiu, H. Zhao, and J. Xiong. 2016. Privacy-aware adaptive data encryption strategy of big data in cloud computing. Paper presented at the 2016 IEEE 3rd International Conference on Cyber Security and Cloud Computing (CSCloud).

Harba, E. S. I. 2017. Secure data encryption through a combination of AES, RSA and HMAC. *Engineering, Technology & Applied Science Research* 7 (4):1781–85. doi:10.48084/etasr.1272.

Hazra, T. K., R. Ghosh, S. Kumar, S. Dutta, and A. K. Chakraborty. 2015. File encryption using fisher-yates shuffle. Paper presented at the 2015 International Conference and Workshop on Computing and Communication (IEMCON).

Jueneman, R. R., D. J. Linsenbardt, J. N. Young, W. R. Carlisle, and B. G. Tregub. 2016. Method for file encryption: Google patents.

Liu, X., G. Yang, Y. Mu, and R. H. Deng. 2018. Multi-user verifiable searchable symmetric encryption for cloud storage. *IEEE Transactions on Dependable and Secure Computing* 17 (6):1322–32. doi:10.1109/TDSC.2018.2876831.

Mahmoud, H., A. Hegazy, and M. H. Khafagy. 2018. An approach for ample data security based on Hadoop distributed file system. Paper presented at the 2018 International Conference on Innovative Trends in Computer Engineering (ITCE).

Maitri, P. V., and A. Verma. 2016. Secure file storage in cloud computing using hybrid cryptography algorithm. Paper presented at the 2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET).

Mazrekaj, A., I. Shabani, and B. Sejdiu. 2016. Pricing schemes in cloud computing: An overview. *International Journal of Advanced Computer Science and Applications* 7 (2):80–86. doi:10.14569/IJACSA.2016.070211.

Panda, M. 2016. Performance analysis of encryption algorithms for security. Paper presented at the 2016 International Conference on Signal Processing, Communication, Power and Embedded System (SCOPES).

Rachmawati, D., A. Sharif, Jaysilen, and M. A. Budiman. 2018. Hybrid cryptosystem using a tiny encryption algorithm and Luc algorithm. Paper presented at the IOP Conference Series: Materials Science and Engineering.

Sajay, K. R., S. S. Babu, and Y. Vijayalakshmi. 2019. Enhancing the security of cloud data using a hybrid encryption algorithm. *Journal of Ambient Intelligence and Humanized Computing* 1–10.

Vishwanath, A., R. Peruri, and J. He. 2016. *Security in fog computing through encryption*. DigitalCommons@ Kennesaw State University.

Yang, L., Z. Han, Z. Huang, and J. Ma. 2018. A remotely keyed file encryption scheme under mobile cloud computing. *Journal of Network and Computer Applications* 106:90–99. doi:10.1016/j.jnca.2017.12.017.

Zou, L., M. Ni, Y. Huang, W. Shi, and X. Li. 2020. Hybrid encryption algorithm based on AES and RSA in file encryption. Paper presented at the International Conference on Frontier Computing.