Implementation of Lashkari et al Scheme and Convergent Encryption on Cloud-Based Secure Deduplication Application

Hermawan Setiawan *, Rico Setyawan
Politeknik Siber dan Sandi Negara, Bogor, Indonesia

* hermawan.setiawan@poltekssn.ac.id

Abstract. The development and growth of data is rapidly increasing and it triggers development on storage technology namely cloud storage. In this research, implementation of Lashkari et al scheme and convergent encryption was conducted on cloud application to provide data confidentiality service and cloud storage space efficiency. Tests were conducted through comparing application that implemented this scheme with application that did not implement scheme within the same condition. Based on statistically significant comparison results, application that implements the scheme reduce 5% of storage space and takes less time for files uploading. Moreover, it can prevent attacker to obtain data illegally by implementing the convergent encryption.

1. Introduction
The development and growth of data increases very rapidly along with the entry of the digital era. The increase is predicted to reach a very fantastic number in 2020 which is as much as 44 trillion gigabytes [1], [2]. The increasing amount of data stored also spurs technological improvements in the field of data storage. One technology that supports data growth is cloud computing which has many benefits including speeding up time [3], low cost, and ease of access [4]. Two problems that exist in cloud storage are data confidentiality [1], [5] and cloud storage resources [1]. The convergent encryption method is an encryption method in the file deduplication scheme that can encrypt / decrypt data copies using the convergent key generated by calculating the cryptographic hash value from the contents of the copy of the data itself [6]. Some researchers have proposed schemes to solve problems in cloud storage systems. The deduplication technique is a special data compression technique to eliminate duplicate data and to optimize the storage space in cloud storage systems [7]. Data deduplication can reduce the memory usage capacity for reserved applications by up to 90-95% and for basic file systems by more than 68% [8] [9]. One study that discusses the file deduplication scheme in cloud storage is the Lashkari et al scheme. This scheme has the advantage of being able to find duplicate data in text files and manage encrypted textual data efficiently [10]. However, the Lahskari et al scheme has the disadvantage of only being applied to text formatted files and does not specifically discuss what techniques are used in the file encryption process. As far as we know, there is also no research that discusses the application of the Lashkari et al scheme in cloud applications. This paper discusses the application of the Lashkari et al and convergent encryption schemes in cloud-based applications that can provide data confidentiality and cloud storage space efficiency services.

2. Method and materials
2.1. Data preparation
The goal of this research is to prove that the application of Lashkari et al scheme and convergent encryption in cloud-based applications can provide data confidentiality services and the efficiency of cloud storage space. This research utilizes 7 data samples consist of raw data, doc, jpg, mp3, mp4, pdf, and ppt. For each type of data, there are 5 approaching size varieties 10 MB; 25 MB; 50 MB; 75 MB; and 100 MB.

2.2. Method
From the literature that has been collected, the scheme is applied to cloud applications. Observations were made on the application to determine the application performance between before and after the scheme was applied. Variables observed include time efficiency and storage space efficiency using
Wilcoxon Signed Rank Test. The applications condition has significant difference if the Sig. value is less than 0.05. Confidentiality testing implements attack scenario to broke the data.

3. Results and discussion

3.1. Implementation of Lashkari et al Scheme on Application

A few modifications were made to the Lashkari et al scheme. This modification aims to adjust the flow of the Lashkari et al scheme and convergent encryption. The original Lashkari et al scheme is shown in Figure 1 and the modified Lashkari et al scheme is shown in Figure 2.

The file hash process is added before the file is uploaded. This is used for the duplication checking process which is done by comparing the hash value of the uploaded file and the hash value of the file that has been stored in the database. The generated hash value is also used as a key in the convergent encryption method. When going through the file duplication check process, files that are not duplicated will be encrypted using a key derived from the file hash value. The encrypted file will then be uploaded to cloud storage and file information will be stored in the database. As for the duplicated files will refer to the previous file that has been stored in cloud storage. Then the file information will be stored in the database.

Implementation is carried out on a laptop with Intel (R) Core ™ i3-2348 CPU @ 2.30GHz processor, 6GB RAM, and 500GB Hard disk. The operating system used is Windows 10. The application is built using the Java language and uses Dropbox as cloud storage and MySQL as a storage database. The file upload menu is shown in Figure 3, the file download menu is shown in Figure 4, and the file delete menu is shown in Figure 5.
3.2. Performance Testing

Performance test used to observe the differences of application before and after implementing the scheme using two variables, storage and time. Application that does not implement the scheme has no file duplication checking process, therefore file will be uploaded entirely to the cloud storage. This condition reduced efficiency of storage space consumption in cloud storage because similar data will be stored within the same time. Table 1 shows the result of performance test towards applications before and after using the scheme.

Table 1. The result of performance test towards applications before and after using the scheme

| Variable   | Condition      | Mean - Std. Deviation | Median – Interquartile | Sig. | Explanation          |
|------------|----------------|------------------------|------------------------|------|----------------------|
| Storage (MB) | Not Implemented | 53,4 – 33,5            | 51,2 – 51,27           | 0,00 | Significant difference |
|            | Implemented    | 39,1 – 25,7            | 37,8 – 43,22           |      |                      |
| Time (sec)  | Not Implemented | 155,3 – 94,8           | 166 – 156              | 0,00 | Significant difference |
|            | Implemented    | 111,4 – 66,9           | 122,5 – 125,5          |      |                      |

The result in the table above shows the comparison of both applications. In general, application that implements the scheme is better in each variable than the other one. It use less storage capacity and processing time. The comparison shows significant differences with deviation value 14,3 MB for
storage space using and 43.9 seconds for time processing. In result, the Sig. value reach 0.00 or <0.005 that indicates significant difference.

| Data Type | Variable | Condition       | Mean - Std. Deviation | Median – Interquartile | Sig.  | Explanation            |
|-----------|----------|-----------------|------------------------|-------------------------|-------|------------------------|
| Storage (MB) Not Implemented | 53.26 – 35.22 | 51.2 – 51.2 | 44 – 31 | 0.005 | Significant difference |
| Storage (MB) Implemented | 41.93 – 25.4 | 44 – 31 | | | |
| Time (sec) Not Implemented | 172.3 – 94.38 | 177.5 – 167.25 | 145 – 119.5 | 0.005 | Significant difference |
| Time (sec) Implemented | 125.3 – 68.24 | 145 – 119.5 | | | |
| Storage (MB) Not Implemented | 53.52 – 35.09 | 51.2 – 51.3 | 38.8 – 28.6 | 0.005 | Significant difference |
| Storage (MB) Implemented | 36.5 – 22.07 | 38.8 – 28.6 | | | |
| Time (sec) Not Implemented | 174.1 – 110.15 | 186.5 – 152.75 | 133.5 - 116 | 0.002 | Significant difference |
| Time (sec) Implemented | 129.3 – 74.72 | 133.5 - 116 | | | |
| Storage (MB) Not Implemented | 53.46 – 34.95 | 51.2 – 51.2 | 35.2 – 16.9 | 0.005 | Significant difference |
| Storage (MB) Implemented | 32.72 – 17.86 | 35.2 – 16.9 | | | |
| Time (sec) Not Implemented | 212.4 – 125.21 | 229 – 209.25 | 169 – 141.25 | 0.009 | Significant difference |
| Time (sec) Implemented | 131.8 – 72.59 | 169 – 141.25 | | | |
| Storage (MB) Not Implemented | 53.26 – 35.22 | 51.2 – 51.2 | 40.4 – 50.7 | 0.005 | Significant difference |
| Storage (MB) Implemented | 44.16 – 32.66 | 40.4 – 50.7 | | | |
| Time (sec) Not Implemented | 147 – 86.11 | 182 – 135.75 | 168 – 131 | 0.005 | Significant difference |
| Time (sec) Implemented | 120 – 74.94 | 168 – 131 | | | |
| Storage (MB) Not Implemented | 52.96 – 35.06 | 51.2 – 49.9 | 47.6 – 49.9 | 0.004 | Significant difference |
| Storage (MB) Implemented | 43.54 – 29.15 | 47.6 – 49.9 | | | |
| Time (sec) Not Implemented | 122.4 – 65.94 | 143.5 – 89.25 | 101 – 77.5 | 0.005 | Significant difference |
| Time (sec) Implemented | 83.5 – 48.48 | 101 – 77.5 | | | |
| Storage (MB) Not Implemented | 52.96 – 35.06 | 51.2 – 49.7 | 31.6 – 28.3 | 0.005 | Significant difference |
| Storage (MB) Implemented | 41.12 – 30.46 | 31.6 – 28.3 | | | |
| Time (sec) Not Implemented | 150.6 – 94.88 | 136 – 178.25 | 71 – 91.25 | 0.005 | Significant difference |
| Time (sec) Implemented | 104.1 – 73.25 | 71 – 91.25 | | | |
| Storage (MB) Not Implemented | 53.36 – 35.33 | 51.2 – 51.1 | 31.1 – 46.4 | 0.005 | Significant difference |
| Storage (MB) Implemented | 33.78 – 24.91 | 31.1 – 46.4 | | | |
| Time (sec) Not Implemented | 108 – 57.86 | 103.5 – 106.5 | 78.5 – 86 | 0.005 | Significant difference |
| Time (sec) Implemented | 85.6 – 52.14 | 78.5 – 86 | | | |

Result for 7 data types testing above shows significant difference on both storage and time variable.

3.3. Confidentiality Testing
Data confidentiality of application can be obtained through utilization of convergent encryption method. Table 3 shows the result of test on data confidentiality of application.
| No | Scenario                                                                 | Result                                                                                           |
|----|---------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|
| 1  | Attacker obtained encrypted file, then decrypt the file using that file.  | Attacker failed to decrypt the file because the hash value used to decrypt did not match          |
| 2  | Attacker obtained encrypted file, then decrypted the file with another file under the same name | Attacker failed to decrypt the file because the hash value used to decrypt did not match            |
| 3  | Attacker input file name that belongs to someone else in file download menu to download the file | Attacker failed to download the file because it was owned by someone else                           |
| 4  | Attacker input hash value of a file with the same name on field file key in file download menu | Attacker failed to download because the file key did not match to the key stored within the database |

### 4. Conclusion

In this research, Lashkari et al scheme and convergent encryption were implemented on cloud application to provide data confidentiality service and cloud storage space efficiency. Tests were conducted through observing four variables namely time efficiency, storage space efficiency, speed and data confidentiality. On performance test, application that implements the scheme is quite better than application that does not implement it, proven by the significant difference value of all variable tested. In the data confidentiality test, application is able to prevent attacker’s attempt to acquire data through implementation of convergent encryption.

### 5. References

[1] L. Wang, B. Wang, W. Song, and Z. Zhang, 2019 *Inf. Sci.* vol 504 pp. 48-60.

[2] M. Zwolenski and L. Weatherill, 2014 *Aust. J. Telecommun. Digit. Econ.* vol. 2 no. 3.

[3] P. R. Kumar, P. H. Raj, and P. Jelciana, 2018 *Procedia Comput. Sci.* vol. 125 pp. 691–697.

[4] N. Vurukonda and B. T. Rao, 2016 *Procedia Comput. Sci.* vol. 92 pp. 128–135.

[5] M. Kavade and A. C. Lomte, 2015 *Int. J. Comput. Appl.* vol. 126 no. 10 pp. 5–9.

[6] J. Li, X. Chen, M. Li, J. Li, P. P. C. Lee, and W. Lou, 2014 *IEEE Trans. Parallel Distrib. Syst.* vol. 25 no. 6 pp. 1615–1625.

[7] R. Kaur, I. Chana, and J. Bhattacharya, 2018 *J. Supercomput.*, vol. 74 no. 5 pp. 2035–2085.

[8] P. Puzio, R. Molva, M. Onen, and S. Loureiro, 2013 *IEEE 5th International Conference on Cloud Computing Technology and Science* pp. 363–370.

[9] J. Xu, E.-C. Chang, and J. Zhou, 2013 *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security - ASIA CCS ’13* pp. 195.

[10] M. Lashkari, A. Suntnure, S. Kathale, dan H. Wankhede, 2019 *International Journal of Research in Engineering, Science and Management* vol. 2 no. 5 pp. 3.