Enhancing Medical Data Security on Public Cloud

Nelson Santos
School of Engineering, Computing and Mathematics
University of Plymouth
Plymouth, United Kingdom
nelson.santos@plymouth.ac.uk

Waleed Younis
Department of Computing and Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
waleed.younis@stu.mmu.ac.uk

Bogdan Ghita
Centre for Security Communications and Network Research
University of Plymouth
Plymouth, United Kingdom
bogdan.ghita@plymouth.ac.uk

Giovanni Masala
Department of Computing and Mathematics
Manchester Metropolitan University
Manchester, United Kingdom
g.masala@mmu.ac.uk

***Abstract*— Cloud computing, supported by advancements in virtualisation and distributed computing, became the default options for implementing the IT infrastructure of organisations. Medical data and in particular medical images have increasing storage space and remote access requirements. Cloud computing satisfies these requirements but unclear safeguards on data security can expose sensitive data to possible attacks. Furthermore, recent changes in legislation imposed additional security constraints in technology to ensure the privacy of individuals and the integrity of data when stored in the cloud. In contrast with this trend, current data security methods, based on encryption, create an additional overhead to the performance, and often they are not allowed in public cloud servers. Hence, this paper proposes a mechanism that combines data fragmentation to protect medical images on the public cloud servers, and a NoSQL database to secure an efficient organisation of such data. Results of this paper indicate that the latency of the proposed method is significantly lower if compared with AES, one of the most adopted data encryption mechanisms. Therefore, the proposed method is an optimal trade-off in environments with low latency requirements or limited resources.**

Keywords—Data Fragmentation, Medical Data Security, NoSQL Database, Cloud Security

I. INTRODUCTION

Cloud computing has transformed the information era, to the extent that increasingly more organisations are adhering to this paradigm, as the ability to deploy and maintain services under a pay-per-consumption model and minimal maintenance costs enables companies to focus on developing and delivering the solution without substantial upfront investments and let the application demands dictate the scaling of the application. However, cloud computing also brings its unique set of challenges, such as the privacy and security concerns emerging from the requirement to outsource user data and applications to the cloud service provider and additional third parties [1]. Often, users rely solely on the cloud service providers to store and safeguard their data, who may choose not to disclose procedures on data storage and handling to clients [2][3]. Cloud service providers (CSPs) have the capability to acquire and access user data stored in their environment, leaving it prone to damage, corruption, or leakage from potential insider threat actors [4][5]. Moreover, given the high number of users using the cloud and the monetary value of their data, attackers frequently target cloud servers and services, leaving cloud service providers to face the huge challenge of dealing with the security and privacy of the data stored in their servers on a high priority basis [5]. Data leakage or loss has a high impact on the trust, business, brand, and reputation of an organisation, all leading to loss of revenue and potential compliance fines [6]. Encryption has been widely implemented to secure data in the cloud; however, encryption methods add additional overhead to computation and users are locked-in by the methods offered by the provider, unless the encryption/decryption process is carried out outside the cloud environment. Further, CSPs often apply general encryption across the entire estate using a company key only known to them, which, although it protects from outside attacks, faces a high risk of insider threat and key theft [4].

This paper proposes a fast and lightweight method for ensuring data confidentiality, which combines data fragmentation with a NoSQL database (MongoDB). The method consists of splitting and randomising data into chunks that are inserted into files that are then stored in a database. In the event that the storage is compromised, attempts on the reconstruction of the original data would be frustrated, even if in possession of the entire scrambled data, as the attacker would not be able to identify the fragmentation pattern. Should the attacker be able to re-assemble part of the data to its original form, it would not be possible to replicate the process across the remaining data, as a new pattern is generated for each file. The proposed method can also be used in conjunction with or as an alternative to other data security methodologies, such as encryption and anonymisation. Its applications range across different scenarios, especially those where speed and confidentiality are paramount, such as medical data, which arguably represents one of the most sensitive, vulnerable, and sought-after targets worldwide [7]. Tampering this kind of data may lead to loss of life. Most hospitals possess a fragile security posture and do not properly address the security of medical images [7]. Having a solution that addresses the security of data in cloud not only enhances the overall security posture of the organisation, but its lightweight and fast nature allows it to be deployed in more specific environments with focused or finite computing resources, such as remote diagnosis, telemedicine, or the public or hybrid cloud data storage, as part of an extension to existing resources.

The remainder of this paper will start with an overview of the state of the art relating to data security in the cloud, following by a detailed explanation of the proposed approach. The paper then outlines a proof-of-concept implementation which was

compared with similar approaches in terms of performance and efficiency. The paper concludes with a summary and critical analysis of the results, along with benefits and the drawbacks of using different techniques to secure data in the cloud.

II. Literature Review

Encryption is the de facto approach used when addressing security and privacy concerns in cloud computing [8]. According to [9] the ideal design for cloud-based cryptographic systems ought to address threats from both potential insider and outsider attacks; in practice most approaches can only deal with one of those sides, typically focusing on outsider threats. The authors in [10] used symmetric tokens to protect sensitive data in the cloud by combing three different encryption algorithms. Similarly, prior research approaches combined symmetric and asymmetric encryption along with hashing and salting to produce a hybrid system that protects sensitive data stored in the cloud [11]. However, combining different encryption techniques results in higher computation overheads, affecting the overall efficiency of the system and does not address the scenario where data is stored in different cloud providers. Sun [9] identifies Attribute-based Encryption (ABE) and Full Homomorphic Encryption (FHE) as the mechanisms that would address both the insider and outsider threats of cloud computing. [12] analysed the feasibility of using ABE to secure medical data in clouds. [13] and [14] introduced a method for checking and verifying outsourced decryption, as both authors claimed it should be a paramount requirement for ABE. However, the design of the key increases in complexity when the attributes in the access strategy set start to grow, leading to additional overhead. Fully Homomorphic Encryption allows operations to be carried out in encrypted data and, when combined with a method designed for outsider threats, it can be a viable option for securing data in cloud. However, FHE has yet to be adopted in practice, despite its advancements, due to the heavy computation overhead and noise creation [15][16].

Another technique, which dates back to the late 70s but is currently witnessing a resurgence in terms of adoption in cloud computing, is data fragmentation [17], as it has significantly lower computational demands to secure the data by relying on concepts from parallel computation. Kapusta and Memmi [18] studied a range of data fragmentation techniques and provided two different categories to represent them. [19] analysed the performance of various fragmentation algorithms and encryption, along with different application scenarios in which the mechanisms would be suitable. [1] proposed a light-weight mechanism based on the chaos system that scrambles the data using a pseudo-random permutation. [20] and [21] uses NoSQL databases to store fragmented data in the cloud. The proposed method takes advantage of the unstructured nature of another NoSQL database to offer data management capabilities and provide a lightweight and faster alternative to securing and ensuring privacy of data in the cloud.

III. Proposed Method

In this section we introduce a novel method used to ensure data security in public clouds using data fragmentation and a NoSQL database. Many systems that hold sensitive data, such as medical images, especially those with specialist functionalities, often have limited resources reserved for additional computing capabilities and security is often an overlooked area. For the proposed method, the following requirements were pursued:

- A balance between resource consumption and security effectiveness must be struck.
- Data residing on a single cloud provider rather than a multi-cloud environment. This is a worst-case scenario as it creates a single point of failure for all the data being stored.

The proposed method, derived from [19], contains the following elements:

- **Chunk**: a variable number of bytes used to divide the original file into sub-array of raw bytes. The size of each chunk can be manually set by the user.
- **Split File**: a special structure that is inserted into the database, composed of a group of randomised chunks concatenated inside. The number of Split Files can be set by the user.

The proposed method uses a random function that calculates the pattern indexes based on a permutation of N elements set by the user and consists of two parts:

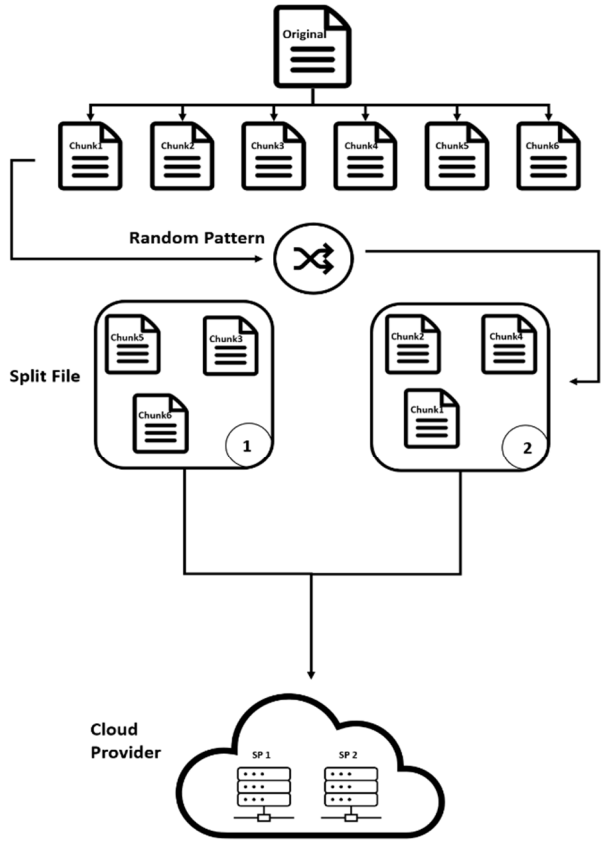

Fig. 1. Random Pattern Fragmentation Disassembly Stage. The original file is fragmented into chunks which are then randomised and stored in split files, which are then uploaded to the database in the public cloud.

- **Disassembly**: As summarised in figure 1, the original file is divided into N chunks and the chunks are inserted into split files in a randomised manner. The length of each split files is proportional to the associated pattern applied, which is set by the user. Unlike other similar approaches, such as [1], the header is not segregated, which hinders even further attempts to reconstruct the file.
- **Reassembly**: As shown in figure 2, the split file is retrieved from the cloud and opened on the client machine. The chunks residing in each split file are rearranged in a dictionary structure according to the pattern of the original files. Once each chunk is reorganised into the original position, the reconstructed file is stored in the client device.

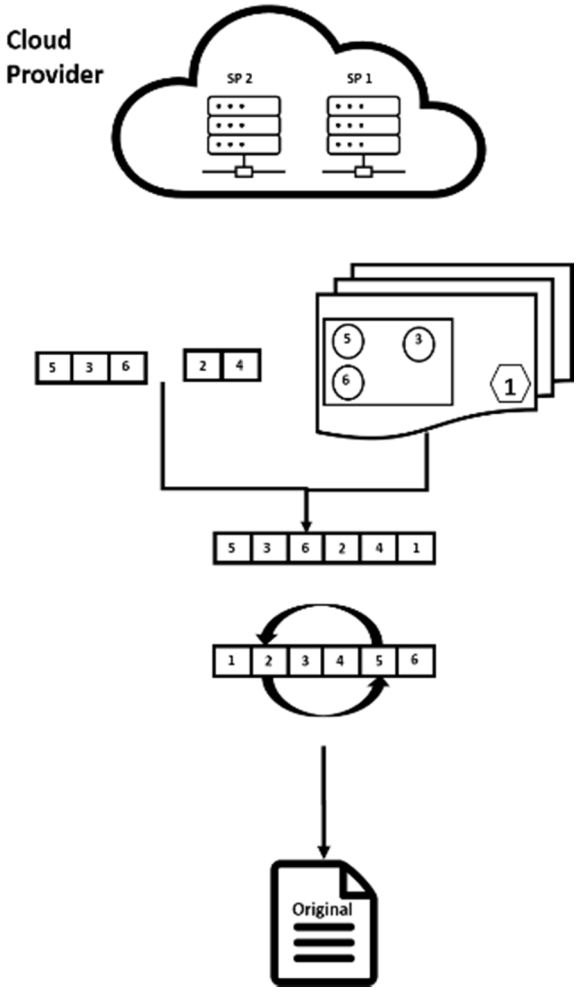

Fig. 2. Random Pattern Fragmentation Reassembly Stage.The split files are downloaded from the cloud server, reorded and reassembled in the client device using the fragmentation pattern. The original file is then restored and saved in the client device.

To add an extra layer of security to the data, the method we propose adds additional steps, which in the disassembly step (figure 3) consists of the creation of point-to-site VPN connection using a Secure Socket Tunnelling Protocol (SSTP) [22] connection, in other words, a TLS-based VPN connection to the Azure cloud infrastructure, where a MongoDB instance resides. Once the connection is established, the Split Files are sent concurrently on different threads to their respective documents to be stored. Each document in the database will account for a single split file, where the randomised chunks will reside within as serialised raw bytes.

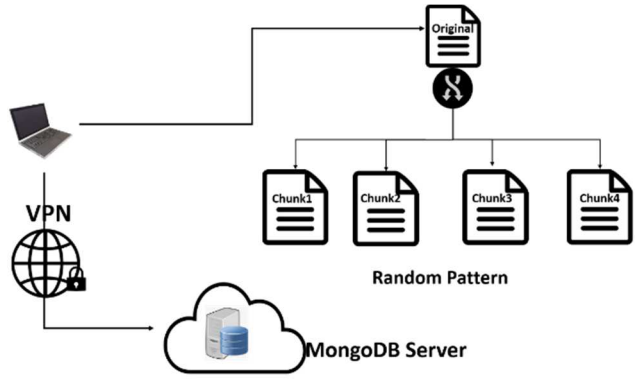

Fig. 3. Proposed Method in the disassembly stage using an SSTP VPN and MongoDB. On the client machine, the random pattern fragmentation is applied dividing the original file into chunks. The chunks are then grouped into Split Files before being inserted on the MondoDB database residing in the cloud.

During the reconstruction phase (figure 4), after the VPN connection is established, concurrent queries are sent from the client to the database, which immediately sends the split files to the client on separate threads. The client then reassembles and de-serialises the chunks into a byte array before storing the file in the original format.

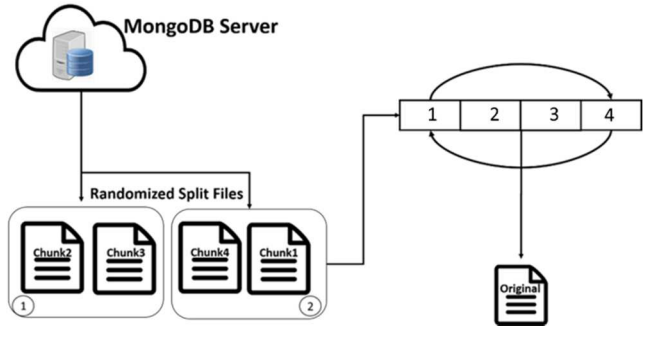

Fig. 4. Proposed Method in the reassembly stage using an SSTP VPN and MongoDB. The Split Files are retrieved from the database, where the chunks are then extracted and reordered into a byte array in the same order as their original format, followed by the restoration of the original file.

A NoSQL database provides more scalability and performance in a cloud environment and its unstructured nature allows data to be stored and retrieved with minimal transformations. MongoDB elevated this feature by introducing support for BSON, a format that supports binary raw files, which allows data to process even faster than other NoSQL databases, such as CouchDB [23]. A VPN connection also ensures that the

data is protected whilst in transit, mitigating therefore the risk of outside threats, more specifically, man-in-the-middle (MIDM) based attacks, where an attacker can intercept the communications and capture sensitive data being transmitted to/from the client, and, in extreme cases, sabotage communications or corrupt the data being transmitted.

IV. Experiment and Results

We tested the performance of the proposed method against other fragmentation algorithms, to determine its latency and determine its applicability in the security of data. The sample dataset used four datatypes (.bmp,.jpeg,.pdf and .docx) with each having 100 KB in size, due to their popularity across multiple environments. Although the program allows the user to set the size of the chunks and the number of split files, for the experiment those values were set to 1000 bytes and 2, respectively. The experiment will consider the total time (latency) taken from the processing of the original file, to its upload on the database, retrieval from the database and reconstruction to its original state.

A MongoDB instance was launched on a virtual private server (VPS) hosted on Microsoft Azure with an Ubuntu 20.04 LTS image. To ensure consistency in the measurements, the database was pre-configured with the required user accounts for the experiment. The proposed method was benchmarked against the implementation of [20], that used random pattern fragmentation (RPF) and a CouchDB (NoSQL) [23] database to split files into chunks and store them, and [21], where the authors proposed a data fragmentation algorithm using a different type of NoSQL (Cassandra) [24]. The device used for the experiment contained an Intel Core I7 -7700 HQ with 2.8 GHz CPU running Windows 10. All methods are based on a similar principle where, after establishing a VPN connection with the cloud, the original file is split into several chunks that are then stored in split files, before being inserted into the database concurrently. Afterwards, the split files are downloaded, and the chunks are then reconstructed back to their original arrangement, before being stored back in the client's machine in the original form. To enhance the security of a file and hinder attackers from reconstructing it, [1] stored the header of the file separately form the other fragments. We decided to test the impact this would have on the latency, by applying the proposed method and separate the header of the .bmp file and storing on a different location in the split file, whilst just applying the proposed method on the .jpg file, the only other image filetype on the dataset. Moreover, given the wide adoption of encryption in the cloud, we also created a program to perform encryption and decryption in the sample data, using AES 256 CBC [25], a symmetric key encryption algorithm that is commonly used to encrypt sensitive data by many industries. This program will allow to compare and contrast its performance against data fragmentation algorithms.

The results from figure 5 indicate a clear distinction between the proposed approach and the approach by [21], with their latency averaging less than half of the other tested methods, whose latency average between around 0.14 and 0.5 seconds respectively. CouchDB has the worst performance of all, with and average of 0.5 seconds, as the support for raw bytes in this platform is limited, requiring additional computational power to process queries. Additionally, the native support for raw bytes offered by Cassandra and MongoDB improves the performance of the overall method. Notably, MongoDB had an even lower latency than Cassandra, with an average latency of 0.15 seconds. There does not appear to be a big performance variation across the different filetypes, which is expected, as the Random Pattern Fragmentation (RPF) algorithm belongs in the bitwise fragmentation category, meaning the fragmentation is independent of file types or associations. However, a performance decrease caused by splitting the header of the .bmp file is evident, as this process has a big impact on the latency.

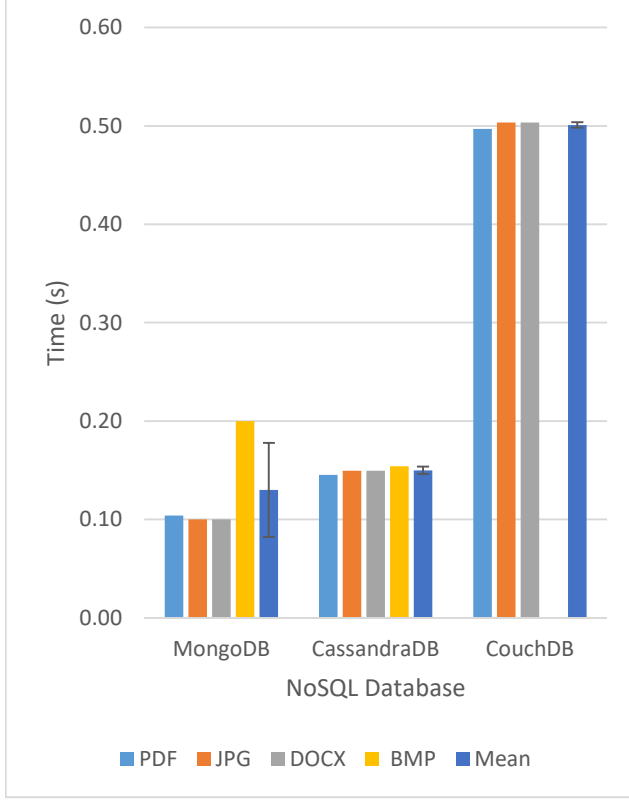

Fig. 5. Performance comparison of the fragmentation sending different format of files in public cloud (Azure) through the Random Pattern Fragmentation on different NoSQL Databases. The Total Time is related to the use of the fragmentation, the transfer time to cloud and the storing in the NoSQL database.

Figure 6 compares the use of the random pattern fragmentation (plus the time related to MongoDB data storage) with respect to AES and the plaintext (the original file without any modifications); AES and plaintext are not implemented in a NoSQL database, therefore is important to note that we are considering the random pattern fragmentation plus the time of MongoDB upload for the proposed method . As expected, the computational overheads brought by AES encryption weight down the performance of this method, rendering it the slowest, taking an average of 0.50 seconds, whilst the proposed method had an average of 0.13 seconds. Nevertheless, the security assurance offered by AES encryption remains higher than the other methods. The proposed method's performance is superior due to its underlying techniques, which rely on processing and uploading the split files simultaneously, as opposed to AES and

plaintext, where the processing and uploading is performed in a sequential manner. The incredibly fast speeds shown by the proposed method also highlight its potential to be applied not only in other industries, but also in environments where real-time data is to be stored and processed, or big data environments.

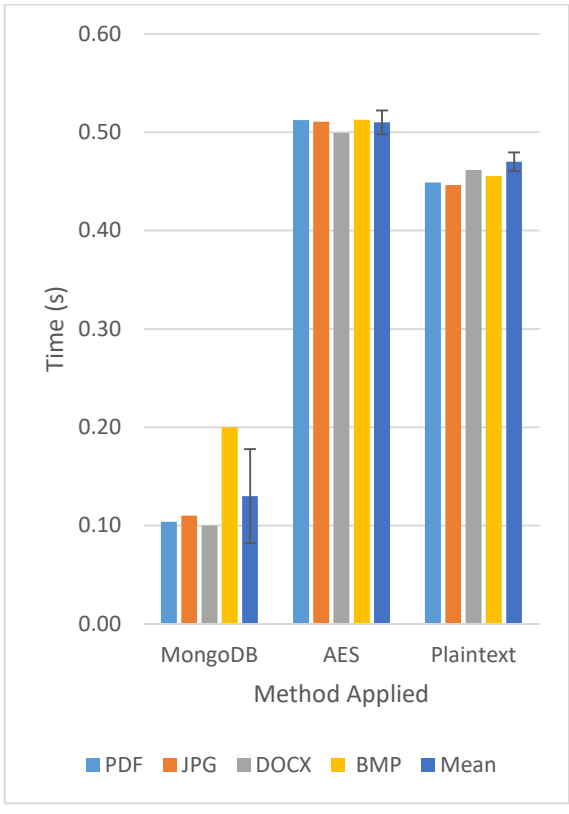

Fig. 6. Performance comparison of the fragmentation sending different format of files in public cloud (Azure) through the Random Pattern Fragmentation against AES and Plaintext. The total time consists of the use of the fragmentation, the transfer time to cloud and the storing in the NoSQL database or the Virtual Private Server.

V. Conclusion

Despite its widespread adoption, data security and privacy remain two areas of high concern in cloud computing. Traditional encryption mechanisms, such as AES, have proven insufficient to cope with the demand of securing data in the cloud due to the nature of unstructured data. Furthermore, the exponential increase of demand in accessing data as fast as possible, hinders the adoption of encryption algorithms, as normally, encrypted data cannot be queried, thus increasing the overhead processing encrypted data. We proposed a lightweight, fast mechanism that secures data in a public cloud using data fragmentation and a randomly generated pattern. By using a NoSQL database to store the data in the cloud, it takes advantage of the unstructured nature of NoSQL to facilitate the management and processing of such data, in addition to the increased support on wider data types to be stored, an important requirement to enhance its adoption to storing medical images in the cloud.

Based on the results, using MongoDB to structure data in cloud based appears to be the best solution associated with the random pattern fragmentation. Indeed, MongoDB has superior performance in comparison to the other possible NoSQL solutions tested, while the random pattern fragmentation guarantees security and fast encoding/uploading in comparison with the AES encryption. Therefore, the method is applicable in all scenarios, in which is important to protect sensitive and varied data and, in particular, it can be adopted for a medical scenario or in all general repositories of data and images. It is a stand-alone method able to protect and archive data that can be used for a private purpose. Naturally, in the case of real applications in a medical scenario, it is necessary to integrate the method with the commonly used health information systems e. g. in PACS (picture archiving and communication system) used primarily in healthcare organisations for medical imaging.

The high performance of this method opens the doors to be applied in environment where speed is paramount, such as big data. Additionally, the flexibility offered by the proposed method, where it can be combined with higher security mechanisms, such as encryption, offers the opportunity to apply this combined method in storage solutions, in environments with higher security requirements. Future work to improve the proposed method would be to apply its capabilities in a big data environment, with a wider variety of file types and an increase in the dataset size. We also plan to investigate the feasibility of the use case where people will need to frequently access data, in order to closely simulate real-world scenarios.

References

[1] M. Bahrami and M. Singhal, "A Light-Weight Permutation Based Method for Data Privacy in Mobile Cloud Computing", *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, 2015.

[2] N. Dahiya, "implementing multilevel data security in cloud computing", *International Journal of Advanced Research in Computer Science*, vol. 8, no. 8, pp. 146-152, 2017.

[3] S. Aldossary and W. Allen, "Data Security, Privacy, Availability and Integrity in Cloud Computing: Issues and Current Solutions", *International Journal of Advanced Computer Science and Applications*, vol. 7, no. 4, 2016.

[4] P. Suwansrikham and K. She, "Asymmetric Secure Storage Scheme for Big Data on Multiple Cloud Providers", *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*, 2018.

[5] S. Chandel, T. Ni and G. Yang, "Enterprise Cloud: Its Growth & Security Challenges in China", *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2018.

[6] R. Rao and K. Selvamani, "Data Security Challenges and Its Solutions in Cloud Computing", *Procedia Computer Science*, vol. 48, pp. 204-209, 2015.

[7] S. Chandel, T. Ni and G. Yang, "Enterprise Cloud: Its Growth & Security Challenges in China", *2018 5th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2018 4th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2018.

[8] Z. Yan, R. Deng and V. Varadharajan, "Cryptography and Data Security in Cloud Computing", *Information Sciences*, vol. 387, pp. 53-55, 2017.

[9] X. Sun, "Critical Security Issues in Cloud Computing: A Survey", *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*, 2018.

[10] Dahiya N. and Rani S. , "implementing multilevel data security in cloud computing", *International Journal of Advanced Research in Computer Science*, vol. 48, no. 8, pp. 146–152, 2017.

[11] Arora A., Khann A., Rastogi A. and Argarwal A. , "Cloud security ecosystem for data security and privacy", in *2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence*, Noida, India, 2017.

[12] M. Li, S. Yu, Y. Zheng, K. Ren and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption", *IEEE Trans. on Parallel and Distributed Systems, vol. 24, no. 1, pp. 131-143, 2013.*

[13] J. Lai, R. Deng, C. Guan and J. Weng, "Attribute-based encryption with verifiable outsourced decryption", *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 8, pp. 1343-1354, 2013.

[14] J. Li, X. Huang, J. Li, X. Chen and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability", *IEEE Transactions on Parallel and Distributed Systems,* vol. 25, no. 8, pp. 2201-2210, 2014.

[15] W. Wang, Y. Hu, L. Chen, X. Huang and B. Sunar, "Exploring the feasibility of fully homomorphic encryption", *IEEE Transactions on Computers*, vol. 64, no. 3, pp. 698-706, 2015.

[16] E. Stefanov, C. Papamanthou and E. Shi, "Practical dynamic searchable encryption with small leakage", *NDSS*, vol. 14, pp. 23-26, 2014.

[17] Shamir A. , "How to share a secret", *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[18] Kapusta K. and Memmi G. , "Data protection by means of fragmentation in various different distributed storage systems - a survey", *arXiv:1706.05960v1*, 2017. [Accessed 14 March 2019]

[19] Santos, N., Lentini, S., Grosso, E., Ghita, B., & Masala, G. (2019). "Performance analysis of data fragmentation techniques on a cloud serve"r. International Journal of Grid and Utility Computing, 10(4), 392-401.

[20] Santos N. and Masala G. , "Big Data Security on Cloud Servers", in *11th International KES Conference on Intelligent Interactive Multimedia: Systems & Services*, Goad Coast, 2018.

[21] N. Santos, B. Ghita and G. Masala, "Enhancing Data Security in Cloud using Random Pattern Fragmentation and a Distributed NoSQL Database", *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, 2019.

[22] "About Azure Point-to-Site VPN connections", *Docs.microsoft.com*, 2020. [Online]. Available: https://docs.microsoft.com/en-us/azure/vpn-gateway/point-to-site-about. [Accessed: 08- Feb- 2021]

[23] "Apache CouchDB", *Couchdb.apache.org*, 2019. [Online]. Available: http://couchdb.apache.org/. [Accessed: 18- Apr- 2019]

[24] "Apache Cassandra", *Cassandra.apache.org*, 2018. [Online]. Available: http://cassandra.apache.org/. [Accessed: 16- Dec- 2018]

[25] "AES — PyCryptodome 3.8.1 documentation", *PyCryptodome*, 2019. [Online]. Available: https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html. [Accessed: 13- Apr- 2019]