A Big Data Storage Scheme Based on Distributed Storage Locations and Multiple Authorizations

Zeyad A. Al-Odat, Eman M. Al-Qtiemat, Samee U. Khan
Department of Electrical and Computer Engineering
North Dakota State University
Fargo, ND, USA
Email: zeyad.alodat@ndsu.edu, eman.alqtiemat@ndsu.edu, samee.khan@ndsu.edu

***Abstract*—This paper introduces a secured and distributed Big Data storage scheme with multiple authorizations. It divides the Big Data into small chunks and distributes them through multiple Cloud locations. The Shamir's Secret Sharing and Secure Hash Algorithm are employed to provide the security and authenticity of this work. The proposed methodology consists of two phases: the distribution and retrieving phases. The distribution phase comprises three operations of dividing, encrypting, and distribution. The retrieving phase performs collecting and verifying operations. To increase the security level, the encryption key is divided into secret shares using Shamir's Algorithm. Moreover, the Secure Hash Algorithm is used to verify the Big Data after retrieving from the Cloud. The experimental results show that the proposed design can reconstruct a distributed Big Data with good speed while conserving the security and authenticity properties.**

***Index Terms*—Big Data, Security, Secure Hash Algorithm.**

I. Introduction

Big Data is defined as a huge amount of different data types that are provisioned through different resources, e.g., social networks, sensor devices, and streaming machines [1]. However, one storage location is unable to handle the increasing size of data; in addition, it is difficult to process Big Data using traditional data processing techniques [2] [3].

Big Data is an emerging technology depending on cloud computing where a massive amount of data is processed [4]. The National Institute of Standards and Technology (NIST) defined cloud computing as a model that enables ubiquitous and convenient network access of a shared pool of configured computing resources [5]. Due to increase of data size, the burden on cloud computing sources has increased. An increase demand of cloud computing causes new methods and tools to be invented. Multiple methods are built for supporting cloud computing in processing the Big Data. For instance, rapid update methods should exist to address cloud storage congestion.

The cloud computing technology becomes one of the main components of information computing technology including data acquisition and retrieving from the cloud. Studies by Cisco and IBM show that 2.5 Quintillion Bytes are generated every day and will reach about 40 Yotta Bytes by 2020 [6] [7]. The cloud computing provides the suitable service to process the Big Data such as Software as Service (SaaS), Platform as Service (PaaS), and Infrastructure as Service (IaaS). All these services are hired individually or mutually to provide quick and efficient data access [8].

The security of Big Data is crucial because a huge amount of data is stored at the same pool of storage location which leads to data interference [9]. Big Data security is guaranteed by different technologies, including the following: 1) Encryption, 2) Centralized key management, 3) User access control, 4) Infusion detection and prevention, and 5) Physical security. Moreover, everyone is responsible for Big Data security, e.g., policies, agreement list, and security software are guaranteed by the cloud service provider [10].

One of the main procedures to store the Big Data is the distribution technology. It divides the big data into parts and distributes them over several storage locations [11]. However, many standards need to be addressed in this scheme including data security and retrieval. The data need to be secured against unauthorized access and protected from data tampering and alteration. Data security is achieved using the Encryption techniques, e.g., Advanced Encryption Standard (AES) while data authenticity is achieved using the Secure Hash Algorithm (SHA). In some cases, attackers can hack the Encryption key and gain access to sensitive data.

This paper proposes a secure and authentic Big Data storage scheme using both Shamir's Secret Sharing (SSS) and SHA. The SSS is used to divide the secret encryption key into parts, this prevents attackers from data decryption even if one part of the encryption key has been obtained. On the other hand, the SHA determines the data integrity through the hash value which is appended with the original data [12].

The rest of paper is organized as follows: Section II provides background demonstrations about the secure hash algorithm and Shamir's Secret Sharing; a literature review is presented in Section III; Sections IV exhibits the proposed methodology; results and discussions are in Section V; Section VI concludes the paper.

II. Background

Before going through the details of our proposal, brief descriptions about the secure hash algorithm and Shamir's Secret Sharing will be presented.

A. Shamir's Secret Sharing

SSS is a method to divide Data (D) into a number of pieces (S_n) where D is easily reconstructable from the minimum

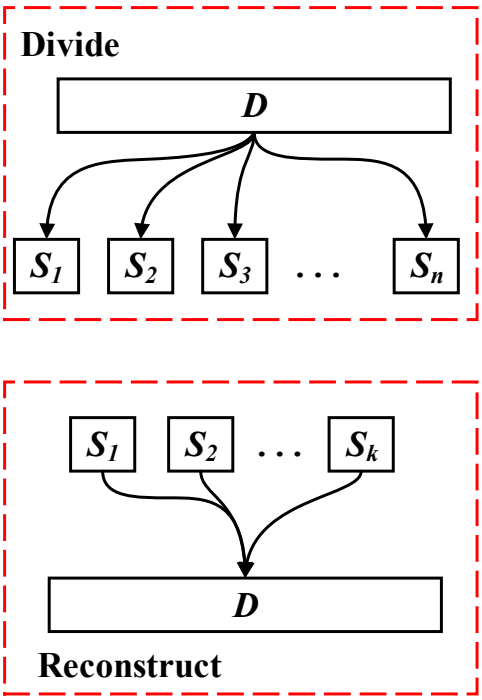

Fig. 1: Shamir's Secret Sharing structure

number of pieces (S_k) [13]. The SSS is a (k, n) based scheme, where k is the minimum number of pieces that are needed to reconstruct the Data (D) and n is the total number of pieces of data (D). However, D is completely undetermined if the number of knowledgeable pieces is fewer than $k-1$. Figure 1 shows the general structure of the Shamir's Secret Sharing, where it involves two operations, the divide and reconstruct. More details will be presented in Section IV.

B. Secure Hash Algorithm

SHA is one of the main cryptography functions. The SHA takes a message (M) of arbitrary size, then through compression function calculations, produces the message hash (H). SHA is used to provide the authenticity and integrity of the data, i.e., ensure that the data are not tampered during transmission or storing.

The secure hash algorithm generates the message hash according to two construction models. The first construction model is Merkle Damgard (MD), which is used to construct the hash functions $MD4$, $MD5$, $SHA-1$, and $SHA-2$ [14]. The second one is the Sponge structure model that constructs the $SHA-3$ hash function [15]. In our proposal, we use the MD structure model to provide data integrity and authenticity. Figure 2 shows the function block of the MD structure model. The message M is preprocessed first by padding the input message to make its size a multiple of block size (B).

In the MD hash standards, the maximum message size that each algorithm accepts is dependent on the block size, where the 512-bit block size accepts messages of size less than 2^{64}-bit, while the others accept a message size of 2^{128}-bit. All hash standards perform the following steps:

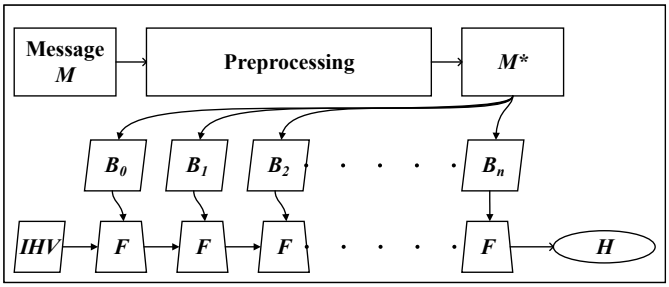

Fig. 2: General structure of the Secure Hash Algorithm (SHA)

1) **Message padding**. In this phase, the message is padded with a sufficient number of zeros to make the message size divisible by the block-size.
2) **Message divide**. After the padding phase, the message is divided into equal size blocks (B) where each size is equal to the desired block size, and each hash standard has a specific block size.
3) **Compression function calculation**. The message's blocks are processed sequentially, one at a time, using the round compression functions according to the selected hash standard. Each block is processed a number of times equal to the number of rounds according to the desired hash function. The output of each block is fed as an input to the second block.
4) **Output hash generation**. After processing all message's blocks, the output hash is taken from the output of the last block.

For more details about secure hash algorithms and their compression functions, the reader is referred to [16].

III. Related Work

The distributed environment of data centers puts an extra burden to the cloud computing technology for processing and retrieving the Big Data. Particularly, the Big Data security is considered as an emerging concern in the area of cloud computing because of the distributed nature of data centers [17]. Cheng *et al.* proposed a Big Data storage scheme based on dividing the Big Data into sequenced parts and storing them among multiple cloud storage locations. The proposed work provides a security protection scheme of mapping between different data elements rather than protecting the Big Data. A trapdoor function is employed to encrypt the storage path of the Big Data which in turn protects the data mapping and reduces the encryption time of the Big Data file [17]. However, the proposed design is vulnerable to cryptography attacks which targets the storage path rather than the Big Data, and once breached the Big Data is easily accessible.

The sensitive Big Data security is presented in different publications [11], [18]–[20], where the cloud service providers have no chance to access the stored data. Li *et al* in [11] proposed an intelligent cryptography approach for secure

distributed sensitive Big Data. The proposed work reduces the chances that cloud operators reach sensitive Big Data. Authors designed a model that securely distributes the Big Data file over multiple storage locations. The Big Data file is divided into parts where each part is encrypted using XOR scheme and sent to the storage locations among the cloud. The proposed design was able to send and retrieve a Big Data file. However, security is related to the security of the Encryption key, and the Big Data file is vulnerable to data modification. In addition, A Security-Aware Efficient Distributed Storage ($SAEDS$) model has been developed in [18] to prevent cloud operators from reaching sensitive data. The algorithm splits the files and saves the data separately in the cloud servers which can significantly increase the scalability of cloud computing in several areas such as the financial industry and governmental agencies. Two algorithms have been proposed to support the $SAEDS$ model: Secure Efficient Data Distributions ($SED2$) is used for data processing prior sending them to the cloud while Efficient Data Conflation ($EDCon$) Algorithm allows users to earn the information by rounding up two data components from distributed cloud servers.

Khan *et al.* proposed a secured Big Data scheme that divides the Big Data file into categories [19]. The proposed design categorizes the Big Data according to its importance, normal and sensitive data. The normal data are stored in the cloud without separation. However, the sensitive data is divided into multiple parts and distributed over the cloud locations. When data are requested, the sensitive data are collected from the storage locations and merged to the normal data and sent to its corresponding user.

Moreover, Dong *et al.* proposed a secure platform to store and share the sensitive Big Data files [20]. The proposed design presents a proxy re-encryption scheme and a process protection method to develop heterogeneous ciphered system functions. In this proposal, authors adopted a process protection technology based on virtual machine monitoring. The virtual machine monitoring is a special key management module that is used to store all public keys that are used in the encryption-decryption process.

A new security framework has been developed in [21] to implement MapReduce tasks on different clusters. The framework allows only a single-sign-on process for jobs submission to G-Hadoop (G-Hadoop is an extension of Hadoop which runs MapReduce tasks on multiple clusters). The proposed model utilizes multiple security solutions such as Secure Sockets Layer (SSL) protocol and a cryptographic mechanism to prevent aggression and misuse of G-Hadoop. This work keeps master-slave architecture of the current G-Hadoop and appending a Certification Authority (CA) server to release proxy and slave credentials.

With the increase of computational power, the Big Data is vulnerable to security breaches, which are not only related to unauthorized access but includes data tampering and sabotage. In this paper, we are presenting a secure Big Data storage scheme based on secret sharing and modification prevention mechanisms.

IV. Proposed Methodology

The proposed design is implemented with the aligning to the architecture shown in Figure 3. To better understand the figure, please refer to Table I that shows the used notations in this section.

TABLE I: Notations

Symbols	Meaning
D	The Big Data
SHA	Secure Hash Algorithm ($SHA-512$)
H	Hash value after applying the $SHA-512$
H^*	Hash value after retrieving D
SSS	Shamir's Secret Sharing
D_n	n part of Data D
E	Encryption Key
E_k	k parts of Encryption key E
$L(x)$	Lagrange polynomial to find $L(0)$
CSP	Cloud Service Provider
AE	Authorized Entity
SE	Service Entity that responsible for Data collection

A. Distribution Phase

In the distribution phase, the Big Data file (D) is hashed using the SHA-512 hash function. The resulting hash value (H) is appended to the file D, as shown in Figure 3. Then, D is divided into parts (D_1, D_2, D_3, ..., D_n), where each part has a unique identification that is needed to reconstruct D. Besides, the hash (H) is appended to the last part of D for storage and encryption. Afterward, the D parts are encrypted using the Encryption key (E) and distributed over multiple cloud locations.

As the Encryption key (E) plays a major rule in the security of D, E is divided into shares (E_1, E_2, ..., E_{x-1}) using the SSS algorithm and distributed over x authorized entities (AEs). According to the SSS algorithm, the number of shares (k), which are needed to reconstruct the secret key, is represented by a polynomial of power ($k-1$), as shown in Algorithm 1.

Algorithm 1: SSS

Input: Encryption Key (E)
Output: E_0, E_1, ..., E_{x-1}

$Determine(k)$; //Least number of shares
for $i \leftarrow 0$ ***to*** $k-1$ **do**
  $a_i = Rand()$
$f(x) = a_0 + a_1x + a_2x^2 + ... + a_{k-1}x^{k-1}$
$Determine(q)$; //Total number of shares
for $x \leftarrow 1$ ***to*** $q-1$ **do**
  $E_x = (x, f(x))$

The algorithm shows the general procedure to divide E into shares (E_{x-1}). First, the least number of shares (k) that are needed to reconstruct E is determined. Then, k random numbers (a_i) are generated to construct the polynomial equation ($f(x)$), where k represents the least number of needed shares to reconstruct the secret key (E). Afterward, all

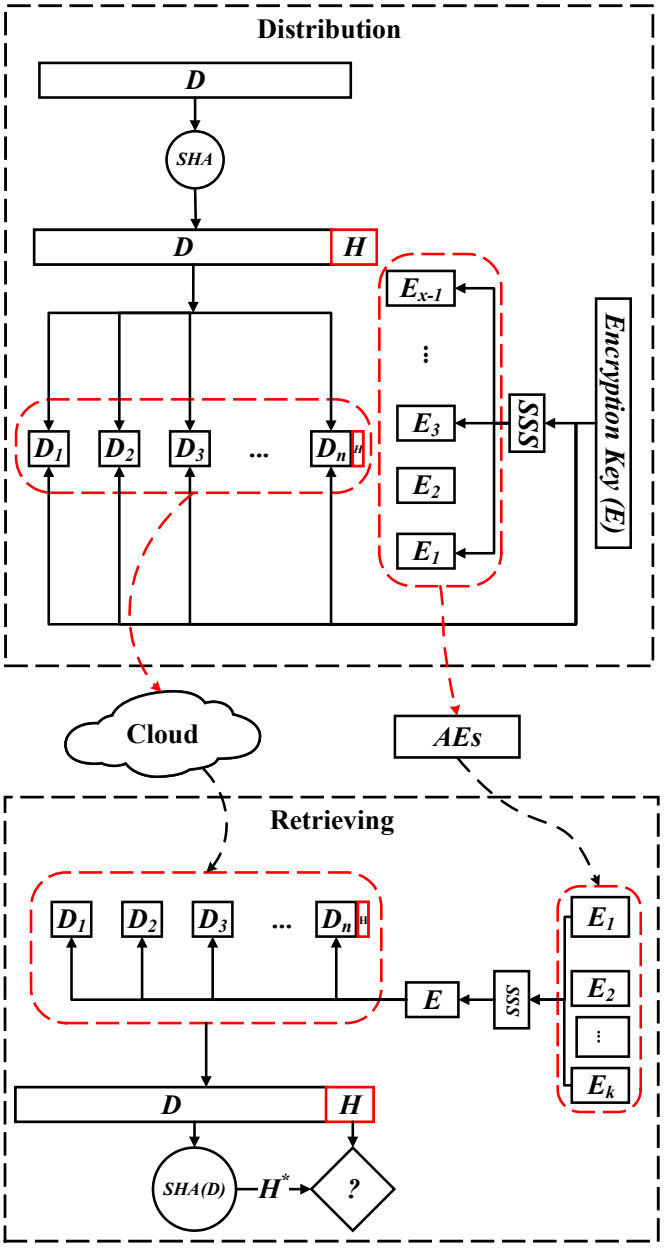

Fig. 3: Schematic diagram of the proposed methodology

shares are constructed using the polynomial equation ($f(x)$) by determining the total number of secret shares (q), where each share (E_{x-1}) is represented as a pair $(x, f(x))$.

B. Retrieving Phase

At the Cloud, the Big Data parts are distributed through multiple Cloud locations where the CSP unable to access any of the D parts because of the data encryption. Moreover, the CSP has no access to the Encryption key because it is divided into shares that are distributed to authorized entities only. In our design, one of the authorized entities (SE) is responsible for the data collection and decryption. However, a single authorized entity is unable to retrieve D because SSS algorithm is used in the design.

To retrieve the Big Data file, half of the AEs need to authorize the data access by giving their E_x where k shares are needed to reconstruct E. Once the least number of E_x shares are collected, then the SE computes E using Equation 1. Afterward, the data parts (D_n) are collected from the storage locations and assembled according to their identifiers.

Equation 1 shows the polynomial calculation procedure to retrieve the Encryption key (E). The equation is called Lagrange polynomials where each point pair $(x, f(x_i))$ refers to one share.

$$L(0) = \sum_{j=0}^{k-1} f(x_j) \prod_{\substack{m=0 \\ m \neq j}}^{k-1} \frac{x_m}{x_m - x_j}, \tag{1}$$

where $L(0)$ represents the retrieved key (E).

After decryption, one more step is needed to verify the integrity of data. The secure hash algorithm is used to accomplish the last step, where the SHA-512 is used to compute the hash value (H^*) of the retrieved data D. Then, the computed hash value (H^*) and the appended hash value (H) are compared to determine whether they are equal. If D is correctly gathered and not modified during transmission, the values of H and H^* are equal. Otherwise, D is corrupted and contains tampered contents.

V. Results and Discussion

The proposed design is tested and verified using sample files of different sizes. The experiments were tested using a configurable experimental environment for large-scale cloud research, Chameleon [22]. All samples and testing results were generated on the Chameleon environment.

A. Experimental analysis

Figure 5 shows the used samples and their corresponding sizes in MegaByte (MB). The Maximum test file size is $\approx 5.2GB$ while the minimum test file size is $\approx 110MB$.

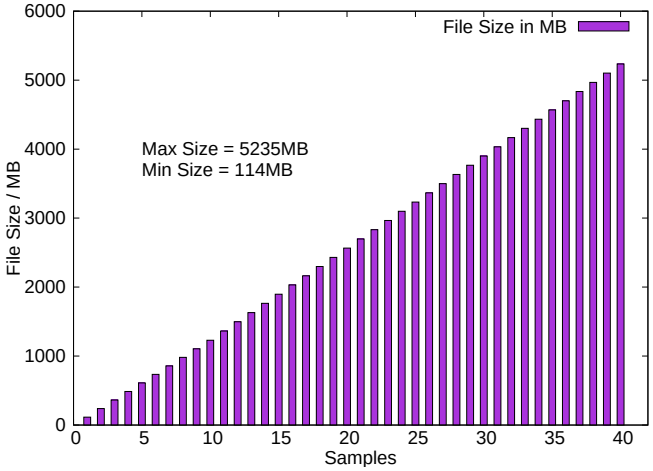

Fig. 5: Size of the sample file measured in MB.

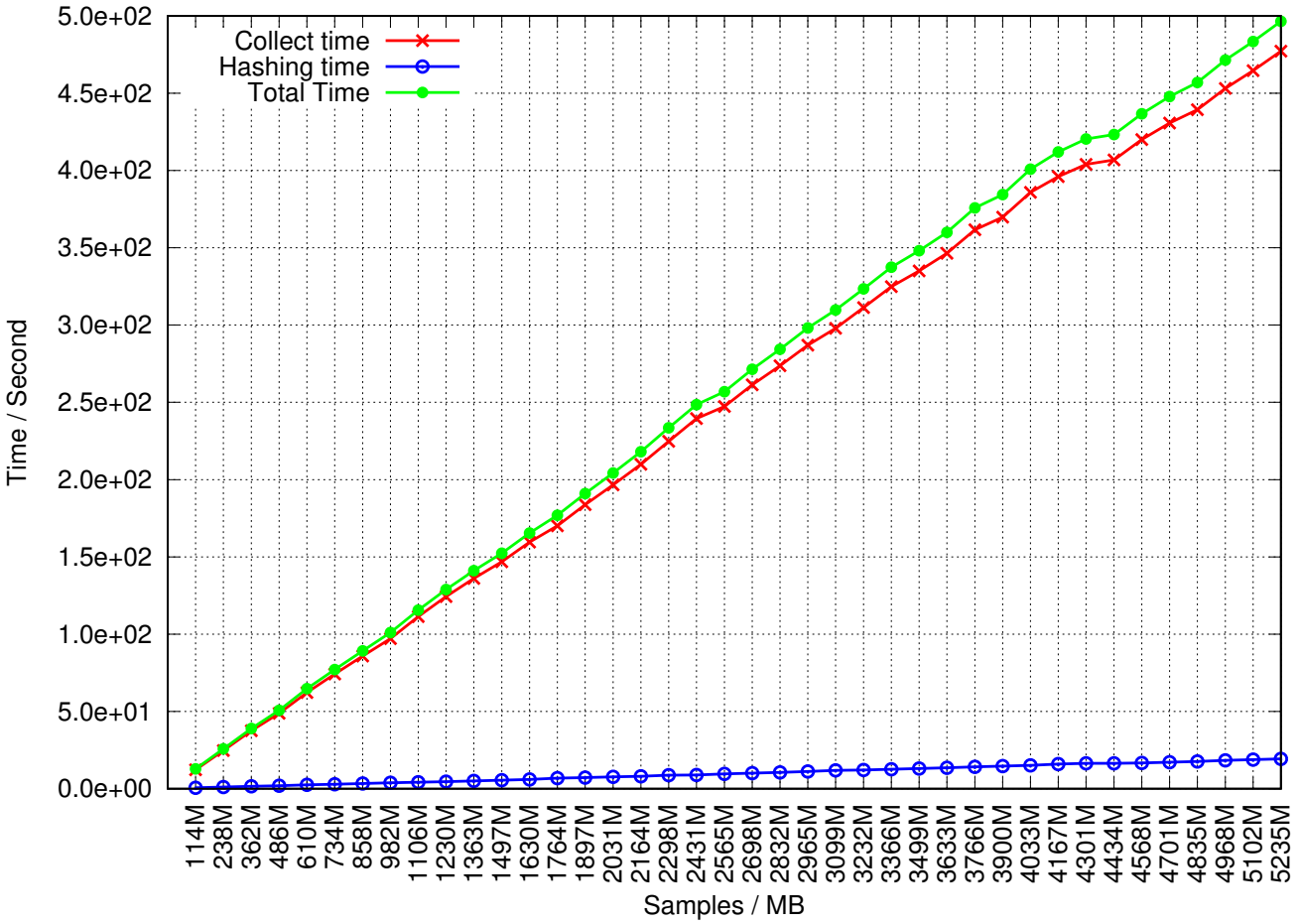

Fig. 4: The time needed to collect and hash the Big Data files.

To test the speed of the proposed design, we measured the elapsed time to collect and hash the test samples, as shown in Figure 4. The figure shows that the hashing time is small when compared to the collection time, which reflects the speed of the hash algorithm in processing the samples. However, the figure shows that the hash time slightly increases when the size of the test file is changed. On other hand, the needed time to collect the data parts is relatively related to the sample size, where they are increasing respectively. Moreover, the increase of data file size affect on the size of each part of the data, which is reflected to the total time to collect and hash the sample files.

B. Security Analysis

We express the security analysis from the Data owner perspective where the security of Big Data is guaranteed by the proposed scheme according to four definitions.

Definition 1. *The Big Data parts* (D_n) *are stored on multiple cloud locations and they are collected by the* SE.

The first level of security is accomplished by the SE, where the distributed data parts are collected and combined together. In addition, the SE acquires one share of the Decryption key.

Definition 2. *The encryption key* (E) *is divided into shares, where each share is given to an authorized entity, and at least half of the authorized entities must provide their secret shares to reconstruct* E.

The second level of security is obtained using the SSS algorithm, which keeps the encryption key secure even if few shares are breached. This property reduces centralized data control by distributing E over multiple users.

Definition 3. *The data integrity and authenticity are conserved by using the* SHA *hash algorithm.*

The third level of security is achieved by using the SHA, where the SHA-512 is employed to ensure that the Big Data file is tamper-free during transmission or storage, and the Big Data parts are reconstructed with the correct order.

Definition 4. *The client concerns about data exposure to the* CSP *is reduced.*

The data owner ensures that the CSP has no access to the stored data. This is achieved by the data encryption and distributed encryption key, where the CSP involvement is only restricted to provide the storage locations. Moreover,

the data parts can be distributed through different $CSPs$, where the SE keeps track of each part using the identification number.

The proposed design is compared with other works with respect to data integrity, CSP prevention, centralized control, and average time to retrieve data from the cloud. Table II shows the comparison between the proposed design and recent works. The proposed design accomplished the security requirements of data integrity and CSP prevention. Moreover, the use of SSS consolidates the proposed design against centralized control.

TABLE II: Comparison with other works

Work	Data Integrity	CSP Prevention	Decentralized Control	Avg Time/ (s)
[20]	✘	✔	✘	-*
[19]	✘	✔	✘	-*
[18]	✘	✔	✘	35E3
Proposed	✔	✔	✔	3.5E2

* Unreported result.

The proposed design outperforms the other works (the reported results) in term of average time needed to retrieve data from the cloud. The employment of the SHA and SSS algorithms consolidates the cloud-based storage system and increases the level of client trust in the $CSPs$.

VI. Conclusions and Future Work

In this paper, a secure storage scheme for distributed Big Data over the Cloud is presented. The proposed design employs the SHA-512 and SSS to ensure Big Data integrity and security. The experimental results show that the proposed design can handle a large data file with a reasonable time while preserving the security and authenticity properties. Moreover, the trust level between the cloud service provider and the client is increasing because of the assured data authorization.

Big Data security is an open scope. In the future, further security analysis will be carried out, in particular, the deployment of secure hash algorithms in future designs.

Acknowledgments

The work of Samee U. Khan was supported by (while serving at) the National Science Foundation. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Results presented in this paper were obtained using the Chameleon testbed supported by the National Science Foundation.

References

[1] C. Thota, G. Manogaran, D. Lopez, and V. Vijayakumar, "Big data security framework for distributed cloud data centers," in *Cybersecurity breaches and issues surrounding online threat protection*. IGI global, 2017, pp. 288–310.

[2] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of âĂIJbig dataâĂİ on cloud computing: Review and open research issues," *Information systems*, vol. 47, pp. 98–115, 2015.

[3] C. Tankard, "Big data security," *Network Security*, vol. 2012, no. 7, pp. 5 – 8, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1353485812700636

[4] Z. Fang, Z. Wu, and L. Luo, "Research on computer information processing technology in the âĂIJbig dataâĂİ era," in *IOP Conference Series: Materials Science and Engineering*, vol. 392, no. 6. IOP Publishing, 2018, p. 062197.

[5] P. Mell, T. Grance *et al.*, "The nist definition of cloud computing," 2011.

[6] R. Pepper and J. Garrity, "The internet of everything: How the network unleashes the benefits of big data," *Global Information Technology Report 2014*, 2014.

[7] S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, and N. Kruschwitz, "Big data, analytics and the path from insights to value," *MIT sloan management review*, vol. 52, no. 2, p. 21, 2011.

[8] M. Abdel-Basset, M. Mohamed, and V. Chang, "Nmcda: A framework for evaluating cloud computing services," *Future Generation Computer Systems*, vol. 86, pp. 12–29, 2018.

[9] C. Tankard, "Big data security," *Network security*, vol. 2012, no. 7, pp. 5–8, 2012.

[10] R. Lu, H. Zhu, X. Liu, J. K. Liu, and J. Shao, "Toward efficient and privacy-preserving computing in big data era," *IEEE Network*, vol. 28, no. 4, pp. 46–50, 2014.

[11] Y. Li, K. Gai, L. Qiu, M. Qiu, and H. Zhao, "Intelligent cryptography approach for secure distributed big data storage in cloud computing," *Information Sciences*, vol. 387, pp. 103–115, 2017.

[12] Z. Al-Odat, M. Ali, and S. U. Khan, "Mitigation and improving sha-1 standard using collision detection approach," in *2018 International Conference on Frontiers of Information Technology (FIT)*. IEEE, 2018, pp. 333–338.

[13] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979. [Online]. Available: http://doi.acm.org/10.1145/359168.359176

[14] I. B. Damgård, "A design principle for hash functions," in *Advances in Cryptology — CRYPTO' 89 Proceedings*, G. Brassard, Ed. New York, NY: Springer New York, 1990, pp. 416–427.

[15] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, "Sponge functions," in *ECRYPT hash workshop*, vol. 2007, no. 9. Citeseer, 2007.

[16] F. PUB, "Secure hash standard (shs)," *FIPS PUB 180*, vol. 4, pp. 1–27, 2012.

[17] H. Cheng, C. Rong, K. Hwang, W. Wang, and Y. Li, "Secure big data storage and sharing scheme for cloud tenants," *China Communications*, vol. 12, no. 6, pp. 106–115, 2015.

[18] K. Gai, M. Qiu, and H. Zhao, "Security-aware efficient mass distributed storage approach for cloud systems in big data," in *2016 IEEE 2nd 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)*. IEEE, 2016, pp. 140–145.

[19] S. Khan and E. Ukey, "Secure distributed big data storage using cloud computing *," *IOSR Journal of Computer Engineering*, pp. 8–12, 2017.

[20] X. Dong, R. Li, H. He, W. Zhou, Z. Xue, and H. Wu, "Secure sensitive data sharing on a big data platform," *Tsinghua science and technology*, vol. 20, no. 1, pp. 72–80, 2015.

[21] J. Zhao, L. Wang, J. Tao, J. Chen, W. Sun, R. Ranjan, J. Kołodziej, A. Streit, and D. Georgakopoulos, "A security framework in g-hadoop for big data computing across distributed cloud data centres," *Journal of Computer and System Sciences*, vol. 80, no. 5, pp. 994–1007, 2014.

[22] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, 2018, vol. 3, ch. 5.