Anonymous Privacy-Preserving Scheme for Big Data Over the Cloud

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

***Abstract*—This paper introduces an anonymous privacy-preserving scheme for big data over the cloud. The proposed design helps to enhance the encryption/decryption time of big data by utilizing the MapReduce framework. The Hadoop distributed file system and the secure hash algorithm are employed to provide the anonymity, security and efficiency requirements for the proposed scheme. The experimental results show a significant enhancement in the computational time of data encryption and decryption.**

***Keywords*—Cloud; Privacy; Security; MapReduce; Big Data.**

I. Introduction

Both cloud computing and big data are two emerging trends in the information technology industries [1]. Cloud computing provides the required services to process and stores the big data. These services encompass, as defined by the National Institute of Standards and Technology (NIST), Software as Service (SaaS), Infrastructure as Service (IaaS), and Platform as Service (PaaS). The big data files become huge and complex because of the variety and size of newly uploaded data to the cloud. For instance, a Cloud Service Provider (CSP) processes many kinds of data structures from different sources, e.g., social network data, medical records, and commercial transactions [2].

To process large and complex data, MapReduce technology is widely used by different entities to process big data. The MapReduce collaborates with cloud computing to produce scalable and efficient platforms. This collaboration leads to a convenient infrastructure that helps the entities to process and store their data effectively and rapidly.

However, the privacy of big data over the cloud becomes a concern because the sensitive information is distributed over various locations and can be retrieved easily. Particularly, when the big data files are not encrypted. This leads to the second issue, which is the encryption and decryption of big data files. It becomes one of the main issues of storing and retrieving of big data file because of the computational time of the encryption and decryption. For instance, a medical record of a patient inside a big dataset requires the decryption of the whole dataset to retrieve the designated record [3].

There are four well-known mechanisms of the privacy-preserving and protection: (i) encryption, (ii) access control, (iii) auditing, and (iv) differential privacy. However, these mechanisms are still considered as open issues in the area of big data and cloud computing. Generally, the uploaded data to the cloud are dynamically updated and not only considered for storage, e.g., online applications. The traditional way to protect data is encryption, which is considered as a challenging task because the majority of applications run on unencrypted datasets. Moreover, in most applications, the data owners and users are different entities that makes the encryption mechanism inefficient to ensure privacy and high-performance systems.

To achieve the goals of: (i) privacy, (ii) access control and (iii) utility anonymization, the data anonymization is considered as a promising approach. However, the infrastructure of data anonymization moves to the MapReduce framework to support large-scale datasets. The framework infrastructure is provided by a Cloud Service Provider (CSP). The CSP provides the required services and management protocols to support the uploaded data with effective cost and flexible usage. However, sensitive information is uploaded to the cloud and exponentially increasing. This information comes from different sources and structures. Therefore, different data-mining techniques were proposed to support the process of data extraction while preserving the privacy.

In this paper, we present an efficient Anonymous Privacy-Preserving Scheme (APPS) for big data over the cloud based on the MapReduce framework. The proposed design helps to maintain data privacy and integrity before they are accessed by the MapReduce. The SHA-512 hash function is employed to provide the integrity requirements for the APPS framework. Moreover, the functional encryption plays a major role in our proposal because it provides on-demand encryption/decryption paradigms.

The rest of paper is organized as follows. Section 2 gives preliminary information related to our scheme. Section 3 provides a literature review of the related works. The proposed methodology is presented in Section 4. Results and Discussion in Section 5. Section 6 concludes the paper.

II. Preliminaries

A. MapReduce

MapReduce is a large-scale data processing framework that allows handling big data in parallel. Google introduced the MapReduce framework in 2004 that got the attention of

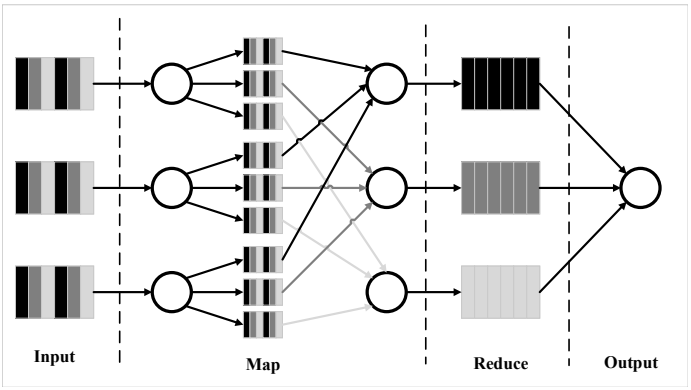

Fig. 1. General structure of the MapReduce functionality

many researchers from different fields. The MapReduce maintains three features: simplicity, scalability, and fault tolerance. Therefore, many entities got benefited from the MapReduce services.

Figure 1 shows the general structure of the MapReduce functionality. The MapReduce consists of two main functions, Map and Reduce. The Map function takes a sequence of values as input and applies a distinct function on each value, i.e., maps an input pair (k_1, v_1) to another pair (k_2, v_2). The reduce function takes a sequence of elements that are linked to the k_2 value and combines them to produce the output pair (k_3, v_3). As shown in Figure 1, the map function maps the corresponding colors from a big list, then the reduce function combines each color to its corresponding group. Afterward, the output will show the results according to the user query.

MapReduce can be implemented on different implementations depending on the environment. For instance, some implementations are suitable for small networks and others are suitable for large connected machines.

B. Functional Encryption

The functional encryption (FE) is defined as an encryption methodology that describes the functions of plain text that can be learned according to a distinct function F [4]. The FE scheme consists of four algorithms, `Setup`, `Keygen`, `Enc`, and `Dec`. The details of each algorithm are shown in Table I. The FE works by generating a pair of public and master keys (P_k, MS_k). Then, for each defined functionality (F), a secret key (S_k) is generated using the MS_k key. A message (m) is encrypted using the public key (P_k) to produce a ciphered message (c). Afterward, if some type of information is requested using function (F), the secret key (S_k) of the function (F) is used to reveal the information from c. Noting that, the decryption is applied to a part of the ciphered text, which saves time in case of big data.

TABLE I
THE FOUR ALGORITHMS OF THE FUNCTIONAL ENCRYPTION

Algorithm	Parameters	Definition
`Setup`	P_k, MS_k	Creates public key (P_k) and master key (MS_k)
`Keygen`	MS_k, F, S_k	Generate secret key (S_k) for the function F using the MS_k
`Enc`	P_k, m, c	Encrypts a message (m) using the P_k to produce encrypted message (c)
`Dec`	S_k, c, y	calculates $y = f(m)$ using the S_k from ciphered data c

C. Secure Hash Algorithm

Secure Hash Algorithm (SHA) is one of the main cryptography functions that compress a message (M) of a predefined size to a fixed size output hash (H). The SHA is used to check the integrity of data, i.e., ensure that the data are tamper-free during transmission or storage. The SHA follows two models to construct the internal compression function operations, Merkle-Damgård and Sponge structures. The MD structure is used to construct the hash functions MD4, MD5, SHA-1, and SHA-2. The Sponge structure is used to construct the SHA-3 hash function. Because of the compression speed of the MD structure, we adopted the SHA-2 hash function in our proposal [5].

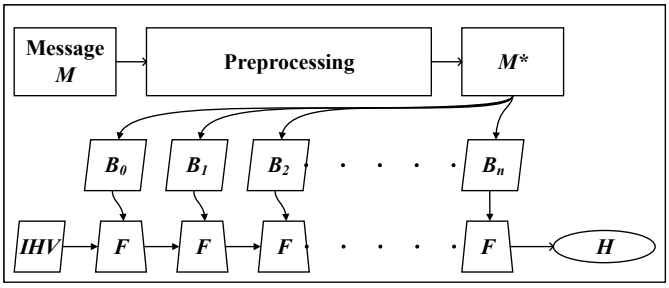

Fig. 2. General structure of the Secure Hash Algorithm (SHA)

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. Figure 2 shows the general operation of the MD structure model, which is performed according to the following steps:

(i) **Preprocessing**. In this phase, the message is padded with a sufficient number of zeros to make the message size divisible by the block-size (B). The message after preprocessing is denoted by M^*.

(ii) **Message divide**. After padding, the message is divided into equal size blocks (B) to make them ready for compression.

(iii) **Compression function calculation**. The blocks are processed sequentially, one at a time, using the compression function (F). Each block is processed a number of times equal to the number of rounds that each function employ. The Initial Hash Value (IHV) is fed as an input to process the first block. Then, the output of processing each block is fed as input to the next block calculation. This process continues until all blocks are processed.

(iv) **Output hash generation**. After processing all message blocks, the hash value is taken from the output of the last block calculation.

D. Threat Model

- Privacy-Preserving. The privacy of the big data is considered a crucial issue, particularly, when all data are stored in the cloud. There should be a high level of trust between the data owner and the CSP.
- Integrity Threat. The shared data in the cloud might be corrupted by an adversary or the CSP loses the stored data due to hardware failure. Moreover, the collision attack forms a serious threat when a weak secure hash algorithm is used [6].
- Encryption/Decryption Time. The time of encryption and decryption need to be enhanced, especially, in the big data environment. As the data are increasing over time, then the encryption will be an obstacle to overcome.

III. Literature Review

The privacy of big data files over the cloud is considered as a crucial issue because of the amount of sensitive information that might be compromised by adversaries. Many researchers in the field of big data security focused on the privacy-preserving of sensitive information over the cloud [7]–[9].

A secure privacy-preserving scheme for on-demand cloud service was proposed in [7]. The proposed work helps to prevent data loss over the cloud and increase the level of trust between the cloud service provider and the data owners. Moreover, the privacy of the stored information over the cloud is maintained and the data owners ensure that the CSP is not able to access the encrypted information on the cloud. the work focuses on the portability of the users information with trusted CSP and store them in locations that are ambiguous to CSP.

Usually, the datasets on the cloud change frequently. Therefore proper privacy-preserving schemes are needed to support the incremental nature of the data over the cloud. Aldeen *et al.* proposed an innovative privacy-preserving scheme for incremental data on the cloud [8]. The proposed design employs an anonymization technique to attain data privacy over distributed and incremental big data. The proposed design works by dividing the data into small parts then stores them on distributed locations over the cloud. The anonymized data are divided according to a predefined anonymization level (k). Then, any new updates to the data will be handled by initializing the anonymized datasets. The evaluation of the design showed enhancements in the execution time of the incremental design over the classical data incremental method.

The privacy of mobile cloud systems is also considered as the same issue as the big data over the cloud. Lo and Gokay prop a hybrid mobile-cloud model based on the concept of cloudlet [9]. Two models were employed to construct the hybrid model, the cooperative and centralized models. One master cloudlet is responsible for the management of the data transfer, ensures the data privacy, and security of communication channels. The proposed design was tested and verified using Mobile Cloud Computing Simulator (MCCSIM) to measure the delay time and consumed power when applying the design.

However, the encryption and decryption operations are considered as part of the dilemma when dealing with big data. Furthermore, the dilemma will increase when considering secure big data schemes with efficient cryptography approaches [10]. To enhance the computational cost of the big data encryption, Li *et al.* proposed an encryption scheme based on Attribute-Based Encryption (ABE) [11]. The proposed design combines the ABE with MapReduce functionality. The encryption of the big data is outsourced to a trusted CSP by maintaining a master node and several slave nodes. The master node provides a set of available slave nodes to perform a specific task. Thereafter, all involved salve nodes are employed as mappers to get the task completed using the MapReduce paradigm. The computational cost, using the proposed design, is reduced into four exponentiations which save more time than the classical encryption method.

The idea of Homomorphic Encryption (HE) for faster encryption of big data was adopted by Wang *et al* [12]. They proposed a faster HE scheme for big data over the cloud. A Fully HE (FHE) along with Dijk, Gentry, Halevi and Vaikuntanathans (DGHV) schemes were employed to improve the encryption speed. The public key size is reduced using Pseud Random Number Generator (PRNG) in cubic form rather than linear form. The security of the proposed design was considered and proved under the error-free approximation problem. Moreover, the system model of the FHE scheme of big data is illustrated.

The anonymous data distribution of big data over the cloud is also adopted by many researchers [13]–[15]. The anonymous data are distributed over the cloud using the MapReduce framework, where the privacy requirements need to be maintained. Zhang *et al.* proposed a privacy-preserving architecture over the cloud using the MapReduce and anonymization functionalities. The proposed design is built on top of MapReduce

framework to achieve the security components of the design. The design support flexible and dynamical data update, and cost-effective framework [13]. The MapReduce framework for data anonymization is also adopted by Thota *et al.* [14]. They presented a centralized and distributed anonymization scheme for healthcare data. Through the design, they propose a new privacy model called LKC-privacy. To further enhance the security of anonymized big data, An enhanced secured MapReduce design is presented in [15]. They propose the Secure Map Reduce (SMR) model to guarantee the privacy and security concerns, which is built as a layer between the MapReduce and Hadoop Distributed File System (HDFS).

In this paper, we propose an Anonymous Privacy-Preserving Scheme (APPS) based on the MapReduce framework. This proposal helps to overcome the security and computational issues that appeared in the area of big data processing. In the subsequent text, we present our proposal and all related design algorithms.

IV. The Proposed Methodology

We present the APPS model based on MapReduce and functional encryption paradigms. In the APPS, the privacy of big data on the cloud is preserved using the functional encryption and MapReduce. Moreover, we employed the SHA-512 hash function to preserve the integrity requirements of the APPS. To save the encryption time, some available nodes on the cloud are incorporated to achieve the encryption task. Before going through the details of our proposal, please refer to Table II to clarify the meanings of the symbols that will be used in the design.

TABLE II
NOTATIONS

Symbols	Meaning
FE	Functional Encryption
SHA-512	Secure Hash Algorithm (*SHA-512*)
CT	Ciphered-Text
MEK	Mapper Encryption Key
MS_k	Master key
P_k	Public key
S_k	Private key
ENC	Encryption
DEC	Decryption
I_{key}	Policy attributes used to generate private key
I_{enc}	Policy attributes to functionally encrypt data
CSP	Cloud Service Provider

A. General Structure

Figure 3 shows the general structure of the proposed scheme. Two levels of MapReduce were utilized to achieve encryption and decryption tasks. Before uploading data to the cloud, the user specifies the number of available nodes in the cloud before uploading the data. The big data file is divided into smaller parts then they are transferred to the predefined nodes (each node is called a *mapper*) for encryption. After encryption, the SHA-512 hash function is applied to each part. After hashing, the Ciphered-Text (CT) of each part is stored in the cloud. On another hand, the MapReduce task is performed to retrieve the requested information from a data part. The data part that contains the desired information is determined first. Then, the integrity of the selected part is examined using the SHA-512 hash function. Afterward, the second level of MapReduce is applied to the selected part. After determining the part that contains the desired information, the FE is used to decrypt the information that is requested by a user. The complete operations are described as follows:

(i) **Divide Phase.** The big data file is divided into smaller parts, where each part is signed using the SHA-512 hash function. The hash value of each part is appended to the end of its corresponding part.
(ii) **Upload Phase.** The data owner uploads the divided data and a set of instructions to the mappers. Each mapper receives part of the data and the encryption instructions to produce functionally encrypted ciphered-text. More details about the instructions are coming through this section.
(iii) **Map Phase.** Each mapper takes attribute instructions and Mapper Encryption Key ($\{MEK\}_{i=1}^{n}$) and produce the functionally encrypted cipher-text (CT_i).
(iv) **Reduce Phase.** The reduce function is performed when a result is requested. The mappers work on the reduce phase to bring the result to a user. then the user needs to decrypt the received result using the associated keys. More details will be elaborated later.
(v) **Integrity Check.** The associated hash value is used to check the integrity of the received data.
(vi) **FE phase.** The desired part of data is revealed using the FE algorithm.

B. System Model

The proposed system comprises the following algorithms:

(i) `Setup(`λ`,`I_{enc}`)`: Number of security parameters (S) are taken as input to produce public (P_k) and Master key (MS_k); a security parameter is denoted by λ. According to a set of encryption policies (I_{enc}), a set of Mappers Encryption Keys ($\{MEk_i\}_{i=1}^{n}$) for n mappers in the cloud are produced.
(ii) `KeyGen(`I_{key}, MS_k`)`: The master key (MS_k) and a set of attributes (I_{key}) are used to generate a private key (S_k) for these set of attributes.
(iii) $ENC(I_{enc}, \{MEk_i\}_{i=1}^{n})$: The encryption policies (I_{enc}) and the $\{MEC_i\}_{i=1}^{n}$ are used to encrypt the message (m). It produces the ciphered-text (CT_i).
(iv) `DEC(CT,`S_k`)`: The ciphered-text (CT) and the secret key (S_k) are used to decrypt the message according to the policy attributes identified by the I_{key} and I_{enc}.

C. Data Management and Anonymization

In our proposal, we used Hadoop data anonymization implementation to deploy our algorithm. Hadoop provides an efficient data anonymization implementation and a reliable infrastructure that processes the data in parallel. In our proposal, five main properties are maintained:

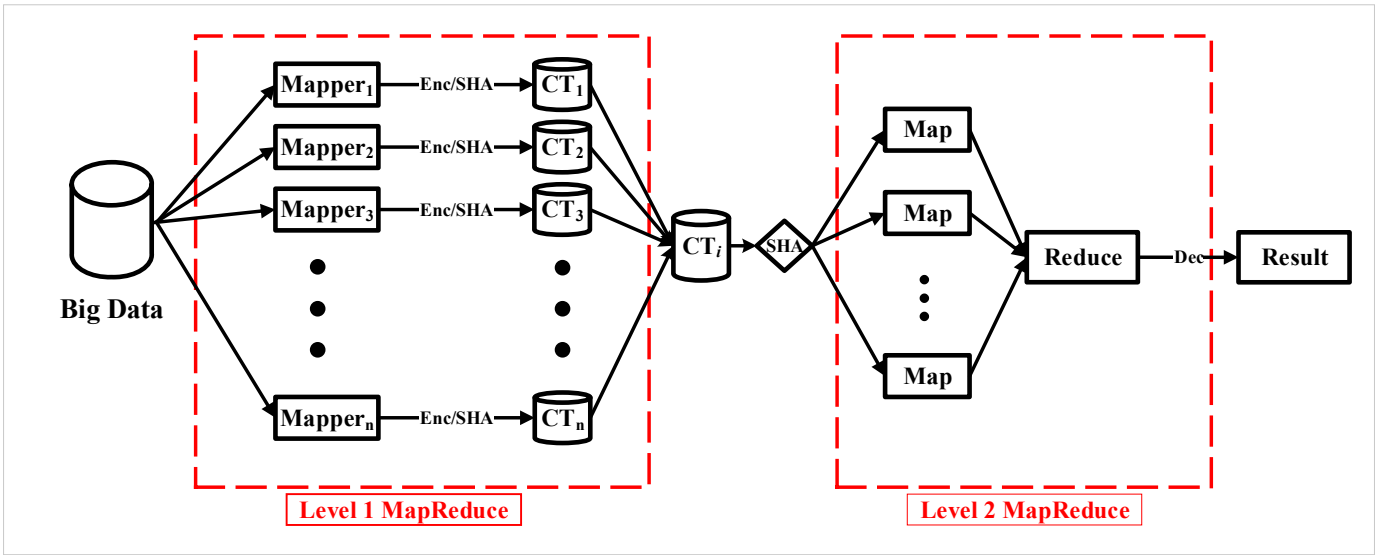

Fig. 3. General structure of the proposed design

(i) data anonymization,
(ii) privacy-preserving,
(iii) data update,
(iv) efficient Encryption/Decryption scheme, and
(v) data integrity.

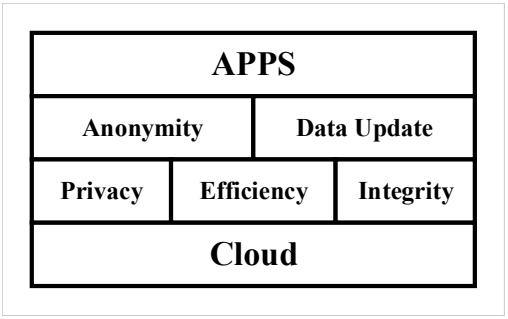

Fig. 4. APPS framework

The anonymized data are kept hidden from the data owners, and only accessed by the MapReduce functionality. The privacy of the data owners is preserved using the utilized anonymization technique, which hides the identity of the data owner. In the case of incremental data, the proposed design supports the addition of new data, which can be added and retrieved efficiently using the MapReduce platform. The time of data encryption and decryption is reduced significantly because the operations of encryption and decryption are delegated to the CSP. The integrity of data is maintained by using the SHA-512 hash function.

D. APPS Construction

Our Anonymous Privacy-Preserving Scheme (APPS) utilizes the MapReduce framework to achieve flexibility and efficiency. To understand the construction model, basic information about Bilinear Map needs to be addressed.

Definition 1: Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_3$ be multiplicative cyclic groups of order p. Suppose g_1 is the generator of $\mathbb{G}_1$, and g_2 is the generator of $\mathbb{G}_2$. Then, a bilinear map e is defined as $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_3$ and satisfies the bilinearity, non-degeneracy, and computability properties.
Each property is defined as follows:

- Bilinearity: $\forall\ u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, then $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degeneracy: $\forall\ g_i\ \exists\ e \Rightarrow e(g_1, g_2, ..., g_i) \neq 1$.
- Computability: $\forall\ u, v \in \mathbb{G}_1$ and $\mathbb{G}_2$ respectively, and $a, b \in \mathbb{Z}_p$. $\exists\ \mathbb{A}\ s.t\ \mathbb{A} \Rightarrow e$

The APPS construction is illustrated as follows:

– **Setup:** The setup stage is executed by the data owners. A bilinear group $\mathbb{G}$ is selected with order p and a generator g. two integers (a, $b \in \mathbb{Z}_p$) are randomly selected and a random oracle is defined using the hash function ($H : \{0,1\}^* \rightarrow \mathbb{G}$). The public and master keys are generated according to the aforesaid setup, where $P_k = (\mathbb{G}, H(.), g, h = g^b, e(g,g)^a)$ and $MS_k = (b, g^a)$.
– **KeyGen(I_{key}, MS_k).** The *KeyGen* algorithm is run for each group of policy attributes (I_{key}). Two random integers are selected (r, r_j) for each attribute j and compute the private key $S_k = g^{\frac{a+r}{b}}, g^r.H(j)^{r_j}, g^{r_j}$.
– **MEKGen**. A randomly, an integer (s) and 1-degree polynomial $R(.)$ are selected, where $R(0) = s$. Produces n splits on s by randomly selecting $s_1, s_2, \ldots, s_n$, where $s = s_1 + s_2 + \cdots + s_n$. Then, the values of MEK are assigned to the values of the s splits ($MEK_i = s_i$, for $i = 1, 2, \ldots, n$).
– **ENC(I_{enc}, MEK)**. The encryption is executed by the MapReduce framework. The encryption policy attributes (I_{enc}) and MEK are uploaded to the cloud, each to a distinct mapper node. Each mapper uses the corresponding MEK to generate the CT_i. Where, for each data part

the output is $Map(I_{enc}, MEK_i) \rightarrow (I_{enc}, CT_{mapper_i})$, which maps between the MEKs and the CTs.

– **DEC**(CT, S_k)**.** When certain data are requested from a user, the desired part of data is retrieved, i.e., (this is done by the set of attributes and data parts pairs, and examined for integrity using the SHA-512 hash function. The functional encryption is applied to the data part, according to the type of requested information, e.g., the user ask only for names and birthdays. Where the retrieved message is retrieved by computing:

$$m = \frac{e(g,g)^{as}}{\frac{e(h^s, g^{\frac{a+r}{b}})}{e(g,g)^{rs}}}.$$

V. Results and Discussion

A. Deployment Environment

We deployed our design based on Chameleon Cloud environment. Chameleon Cloud is a configurable experimental environment for large-scale cloud research at the University of Chicago and the University of Texas at Austin. The structure of the deployment environment is depicted in Figure 5. A Kernal-based Virtual Machine (KVM) virtualization is installed on top of Linux (Ubuntu LTS 16.0.4) operating system and the reserved hardware. Hadoop is installed via OpenStack to facilitate the intensive MapReduce operations.

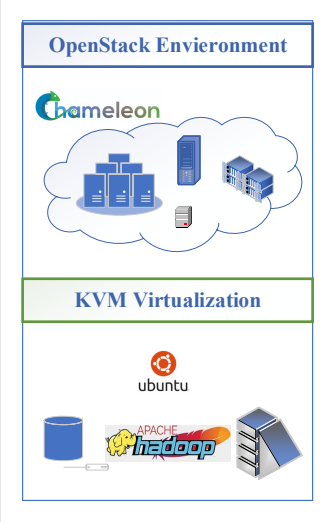

Fig. 5. Deployment environment structure

B. Experimental Test

To test the validity of our approach, the ChameleonCloud environment is utilized with the following setup:

(i) We generate synthesized datasets with different size.
(ii) Utilize Hadoop framework by deploying the Hadoop Distributed File System (HDFS) appliance that is provided by the ChameleonCloud environment.
(iii) The OpenStack environment is used to manage the virtual machine environment and resource scheduling.
(iv) The KVM virtualization software is used to provide unified storage resources. This is provided by deploying the MVAPICH appliance that is provided by Chameleon, which is responsible for MPI clustering of the KVM virtual machine with InfiniBand enabled.
(v) Utilize the Ubuntu 16.04 LTS appliance, which is supported by Chameleon.
(vi) The size of the synthesized datasets is ranging from 200 MB to 4.0 GB. The execution time is measured to evaluate system efficiency.

The proposed design is compared with the centralized anonymous approach by Mohammed *et al.* and privacy scheme by Zhang *et al.* Figure 6 shows the comparison between our proposal and the other approaches. In our design, we included higher data sizes than the data presented in the other works. The centralized approach shows a good start in the execution time with lower size, but a significant increase in the execution time when the dataset's size slightly increase. Approximately, a linear increase in the execution time for our proposal and the work of Zhang *et al.* with a noticeable advantage of our work in the analogous range of dataset's sizes. Furthermore, our work has been tested on higher synthesized datasets' size than the other works.

C. Final Remarks

The security of the proposed scheme is analyzed according to the following threats.

(i) Data Integrity. The APPS model is secure against data integrity and authenticity issues. The secure hash algorithm SHA-512 ensures that the data are not tampered or modified during storage or transmission.
(ii) Privacy-Preserving. The APPS model guarantees the privacy of the uploaded data. Furthermore, if one of the mapper nodes is curious, the model still able to preserve the privacy of the uploaded data as well as the data owners identities. This is held according to the Bilinear maps assumption.
(iii) The Encryption/Decryption time is reduced significantly, as shown in the experimental results. Moreover, the FE showed the ability to decrypt part of data in case of a specific type of information is required.

VI. Conclusions and Future Work

This paper presented an Anonymous Privacy-Preserving Scheme (APPS) for big data over the cloud. The proposed design employed the SHA-512 hash function, the functional encryption algorithm, and the MapReduce framework. Through two levels of MapReduce, the proposed design enhance the encryption/decryption task of big data. The employment of the functional encryption helped to decrypt the desired part of data rather than all data, which saves time and efforts. The Integrity of the data is conserved using the SHA-512 hash function and the privacy of the uploaded data is conserved using the anonymous MapReduce framework.

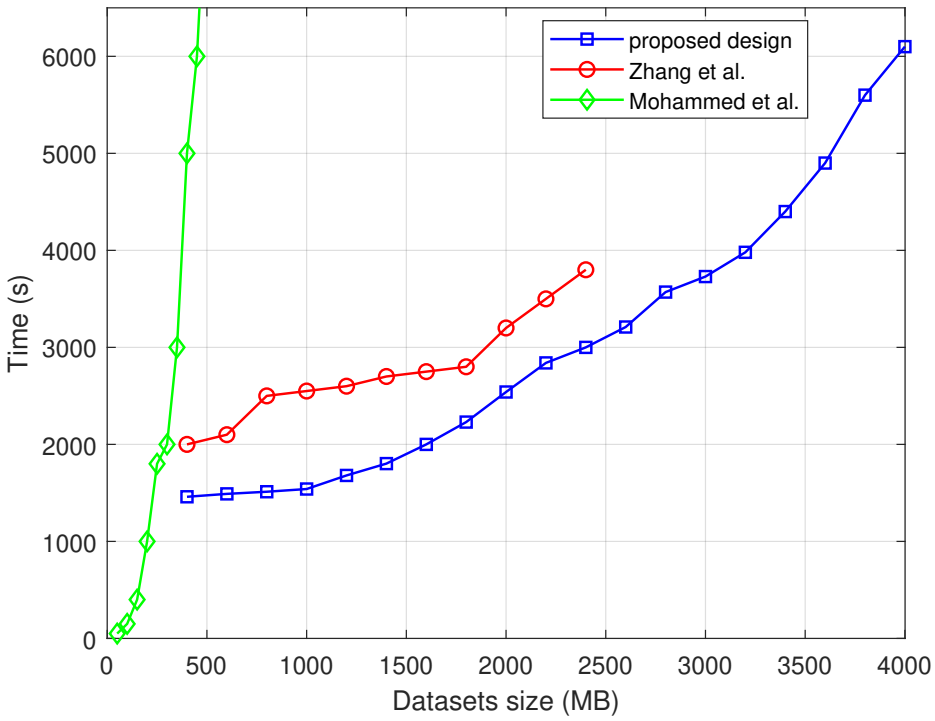

Fig. 6. Execution time of the proposed design and other works

In the future, further experiments will be conducted to evaluate the performance of the proposed scheme. More security issues and threats will be analyzed in the future.

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. Yang, Q. Huang, Z. Li, K. Liu, and F. Hu, "Big data and cloud computing: innovation opportunities and challenges," *International Journal of Digital Earth*, vol. 10, no. 1, pp. 13–53, 2017.

[2] Z. A. Al-Odat, S. K. Srinivasan, E. M. Al-Qtiemat, and S. Shuja, "A reliable iot-based embedded health care system for diabetic patients," *arXiv preprint arXiv:1908.06086*, 2019.

[3] Z. Al-Odat, E. Al-Qtiemat, and S. Khan, "A big data storage scheme based on distributed storage locations and multiple authorizations," in *2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. IEEE, 2019, pp. 13–18.

[4] Z. Brakerski and G. Segev, "Function-private functional encryption in the private-key setting," *Journal of Cryptology*, vol. 31, no. 1, pp. 202–225, 2018.

[5] Z. Al-Odat and S. Khan, "The sponge structure modulation application to overcome the security breaches for the md5 and sha-1 hash functions," in *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 811–816.

[6] 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.

[7] D. Chandramohan, T. Vengattaraman, and P. Dhavachelvan, "A secure data privacy preservation for on-demand cloud service," *Journal of King Saud University-Engineering Sciences*, vol. 29, no. 2, pp. 144–150, 2017.

[8] Y. A. A. S. Aldeen, M. Salleh, and Y. Aljeroudi, "An innovative privacy preserving technique for incremental datasets on cloud computing," *Journal of biomedical informatics*, vol. 62, pp. 107–116, 2016.

[9] A. T. Lo'ai and G. Saldamli, "Reconsidering big data security and privacy in cloud and mobile cloud systems," *Journal of King Saud University-Computer and Information Sciences*, 2019.

[10] P. P. Sharma and C. P. Navdeti, "Securing big data hadoop: a review of security issues, threats and solution," *Int. J. Comput. Sci. Inf. Technol*, vol. 5, no. 2, pp. 2126–2131, 2014.

[11] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *International Conference on Information and Communications Security*. Springer, 2012, pp. 191–201.

[12] D. Wang, B. Guo, Y. Shen, S.-J. Cheng, and Y.-H. Lin, "A faster fully homomorphic encryption scheme in big data," in *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(*. IEEE, 2017, pp. 345–349.

[13] S. Madan and P. Goswami, "A privacy preserving scheme for big data publishing in the cloud using k-anonymization and hybridized optimization algorithm," in *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*. IEEE, 2018, pp. 1–7.

[14] C. Thota, R. Sundarasekar, G. Manogaran, R. Varatharajan, and M. Priyan, "Centralized fog computing security platform for iot and cloud in healthcare system," in *Fog Computing: Breakthroughs in Research and Practice*. IGI global, 2018, pp. 365–378.

[15] P. Jain, M. Gyanchandani, and N. Khare, "Enhanced secured map reduce layer for big data privacy and security," *Journal of Big Data*, vol. 6, no. 1, p. 30, 2019.