JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 33, 713-725 (2017)
DOI: 10.6688/JISE.2017.33.3.7

SiDS: Sudoku Inspired Data Scrambling Method for Securing Data in Cloud using *N*-cloud Architecture

K. S. VIJAYANAND AND T. MALA
Department of Information Science and Technology
College of Engineering Guindy
Anna University
Chennai, 600025 India
E-mail: ksvijayanand@gmail.com

Cloud computing has become one of the leading computing paradigms in the current era. Most of the enterprises, business people, academia, and even individuals use the cloud as a convenient platform for storing their data. Since cloud incorporates many computing technologies, the security issues associated with those technologies will be carried over to the cloud too. Loss of control over the data stored in public Cloud Service Providers (CSPs) and the lack of trust in the service providers increase the risk of storing data with public CSPs. The commonly used solution for data security is encryption, but most of the cases it consumes a large amount of time. More than that, the size of the data will be increased when it is encrypted. Since the whole data is encrypted and stored in a public CSP for a long time, the security of the data depends on the trustworthiness of the CSP and the strength of the encryption algorithm. In this paper, a Sudoku puzzle inspired Data Scrambling method (SiDS) is proposed to improve the data security using data splitting and *N*-Cloud architecture. The SiDS provides fast and efficient data splitting mechanism, by which the data is split into multiple chunks and stored into *N* number of CSPs. The use of *N*-cloud architecture eliminates the vulnerability of single-point-failure. The performance of the proposed system is analyzed with various security metrics and found giving better results, and the execution time of the proposed algorithm consumes less time than various block cipher encryption algorithms.

Keywords: cloud computing, data security, data splitting, sudoku puzzle, *N*-cloud architecture

1. INTRODUCTION

The National Institute of Standards and Technology (NIST) gives a brief and particular definition for cloud computing as a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction [1]. Depend on the requirements and constraints, cloud services can be availed in different deployment models like Private Cloud, Public Cloud and Hybrid Cloud [2]. These cloud deployment models provide storage as a service as one of their services. When data is stored within Public CSPs, security risk will be more. The depth and width security and privacy issues with cloud data services are discussed in [3]. At present, due to the advancement of technology it has resulted in the outbreak of data volume in all business sectors. The organizations started to use the cloud for keeping large data where sensitive data is stored in private side itself and non-sensitive data in the cloud. It is very difficult to manage the big data within private side itself. More than that, separating sen-

Received July 15, 2016; accepted September 21, 2016.
Communicated by Balamurugan Balusamy.

sitive and non-sensitive data is a challenging task. The organizations, who handle third party client's data, prefer public CSPs when they need to store huge volume of data.

Since the data is stored in a public CSP, the private data is under the risk of being accessed by unauthorized entities. Even though the Public CSPs have advanced security measures, as of now, no mechanisms are full proof. No Public CSP will guarantee the complete security for the data being stored with them. Even a detailed Service Level Agreement (SLA) [4] is framed between the customers and the CSPs, the loss is only for the customers if any security breach occurs. The only option for the customer is to trust the public CSPs and their security measures.

The customer who uploads their private data don't want any unauthorized entities access or read or manipulate their data. Unauthorized entities can be a lethal hacker, a competitor, and sometimes the CSP itself [5]. Even though the data is not lost but read by someone else may cause serious damage if the data contains very sensitive and critical information. Hence, the customers want their data should be in unreadable and non-understandable format when it is stored in public CSP. The common method to secure data as said above is cryptography. The power of any cryptographic method lies with the power of encryption and decryption algorithms. As the strength of encryption algorithm increases, the computational complexity also increases. Many cryptographic algorithms are evolved over time, but none is claimed as unbreakable. In the case of the data which is encrypted and stored with public CSPs for long term storage, if the CSP itself is the attacker more chances are there for the data to be revealed. Another drawback with cryptosystems is that the size of encrypted data will be more than that of the original data, which can be an overhead as the payment is based on the size of the data to be stored.

Our work aims to overcome the above-mentioned scenarios. Just because the entire data is encrypted and stored with one CSP, the CSP may get access to the data if the encryption algorithm fails. To avoid this, we use multiple CSPs for storing the data. The data is fragmented into multiple chunks and stored into multiple CSPs. Each chunk by its own does not infer any information. The size of the chunks should be small, and hence the total number of chunks may be large. Only if all the chunks are identified and fused together in the proper order, the data become meaningful. The challenge for the attackers will be identifying the various chunks and fusing them together.

The data splitting concept was mainly used in the computer paradigm for improving the access mechanisms and storage purposes. Some researchers have used the concept for secure data storage in distributed environments. In recent times, some research papers have proposed different frameworks which use separation of application and data, and fragmentation of data. They have used the *N*-cloud architecture to store multiple data fragments or data and applications. In the existing data splitting works, the data are divided into multiple fragments in a sequential fashion. A certain number of fragments selected sequentially or randomly will be stored into multiple CSPs. Some sort of identification mechanisms, like sequence numbers, will be used for re-assembling the fragments. But, such identification mechanisms may give clues for the attackers to track the various fragments. Hence, in this proposed work, we use sudoku inspired data scrambling method to overcome such limitations. The rest of the paper is structured as follows. Section 2 discusses the related works and the state-of-the-art security mechanisms for cloud storage. Section 3 gives the description of the proposed approach, SiDS. The security and the performance of the system are analyzed in Section 4, followed by conclusion in Section 5.

2. RELATED WORKS AND THE STATE-OF-THE-ART SECURITY MECHANISMS FOR CLOUD STORAGE

A. Multi-cloud Architecture (*N*-cloud Architecture)

As cloud technology is evolving more and more, so does the need for security of data stored remotely in the cloud setup. It is challenging to secure the data that is stored in a public cloud open to attacks. A solution is proposed where four distinct architectures that allow multiple distinct clouds to operate simultaneously are discussed and their merits are evaluated [6, 7]. They are (i) Replication of applications: This allows the client to receive multiple results from one operation performed in distinct clouds and to compare them within their own premises and enables the client to get evidence on the integrity of the result; (ii) Partition of application system into tiers: This system allows separation of the logic from the data and gives additional protection against data leakage due to flaws in the application logic. This system tries to reduce the data leakage by separating the application logic into tiers; (iii) Partition of application logic into fragments: Here, the system allows distributing the application logic to distinct clouds. It has two benefits. First, no cloud provider learns the complete application logic. Second, no cloud provider learns the overall calculated result of the application. Thus, this leads to data and application confidentiality; (iv) Partition of application data into fragments: Data partition system allows distributing fine-grained fragments of the data to distinct clouds. None of the involved cloud providers gains access to all the data, which safeguards the data's confidentiality.

The proposed system tries to incorporate the fourth architecture that allows the application data to be split into fragments and placed in different cloud providers.

B. Data Partitioning Techniques

In a hybrid cloud environment when the private organization opts for storage at a public CSP, it is essential to ensure that the data at CSP is secure from threats. This is done by partitioning each tenant's data and encrypting these partitions twice before placing them into separate partition tables at the cloud service provider. In the work mentioned in [8] each tenant's data is partitioned into separate partitions. This is followed by a technique called DPET (Data Partition Encryption Technique) which encrypts the tenant data twice.

The proposed system uses this concept of encrypting the partitions before placing them in different cloud service providers.

C. Encryption to Improve Data Security

As processing at the CSP is efficient in this storage scenario, there is a need for a mechanism to improve security and data integrity in the cloud by using data partitioning technique and various data encryption techniques like DES, AES, MD5 and RSA algorithms [9, 10]. There exist several architectures that combine recent and standard cryptographic primitives in order to achieve the goal of building a secure cloud storage service on top of a public cloud infrastructure [11]. Another alternative solution was proposed by database as a service to ensure the data privacy based data encryption [12]. Each encryption algorithm has their own merits and demerits and is used in the contexts that suit for its merits.

D. Homomorphic Encryption Technique

A fully homomorphic encryption technique, which allows the service provider to process the data in encrypted form itself, is the next step towards improving data security in a hybrid setup [13]. Homomorphic encryption reduces the security risk by avoiding the need of decrypting the data for performing operations. But, so far, this mechanism supports very limited operations only. The research community is looking forward to an improved encryption mechanism with less computational overhead [14].

In our previous work [15], a method was proposed for data security in the cloud using data splitting followed by encryption with *N*-cloud architecture. In this paper, the work has been done in the area of data splitting and scrambling based on Sudoku puzzle concepts for providing data security in cloud, which consumes fewer resources than conventional cryptographic algorithms.

3. SUDOKU INSPIRED DATA SCRAMBLING

Here, an approach is proposed for scrambling the data using sudoku puzzle [16-20]. The original data, that to be stored in the public cloud, is split into multiple chunks. All these data chunks will be virtually embossed into a solved sudoku puzzle table. The numbers in the sudoku table refer to the identifier for CSPs. The proposed approach is explained in detail, in the following sections.

3.1 The Framework

The framework shown in the Figs. 1 and 2 are generic frameworks for the Sudoku Inspired Data Scrambling Method (SiDS) proposed here. The framework gives a graphical representation of the flow of the processes and the methods included in the SiDS. The Fig. 1 shows the processes for data splitting, scrambling and uploading, whereas the Fig. 2 shows the processes for data downloading, unscrambling and fusing together.

A sudoku puzzle table, of size *N***N*, is generated using a generator function. This generator function can generate the same sudoku puzzle at any time, using the same seed value. The generated sudoku table will be solved, then after. The solved table will contain *N* number of boxes, each box will contain 1 to *N* number of elements in each row and column, and thus totally *N***N* number of elements will be displayed. For eg., in the case of a 9×9 sudoku table, each row and column will contain numbers ranging from 1 to 9, and the table contains nine 3*3 sub-grids, where each sub-grid contains 1 to 9 numbers. Thus, the whole table contains 81 elements.

In this work, each element in the solved sudoku table represents the identifier of public CSPs. In the case of an *N***N* table, *N* number of CSPs will be used for data storage purpose and each CSP will be given identifiers (CSP ID#) starting from 1 to *N*. A separate queue will be maintained in the private cloud for each CSP. The original data, to be stored, is split into *N***N* number of chunks, and each chunk is virtually placed over the Sudoku.

The scenario is illustrated here with the example sudoku table given in the Fig. 3. In Fig. 3, a solved sudoku puzzle is shown. The elements in the table indicate the CSP ID#. The data chunks are logically placed on the table in a row-wise; *i.e.* first 9 chunks are

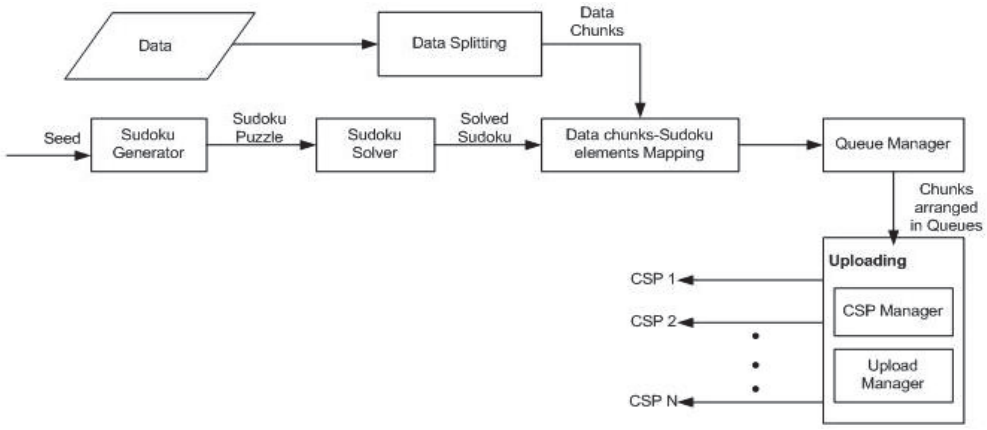

Fig. 1. Framework for data chunking, scrambling and uploading.

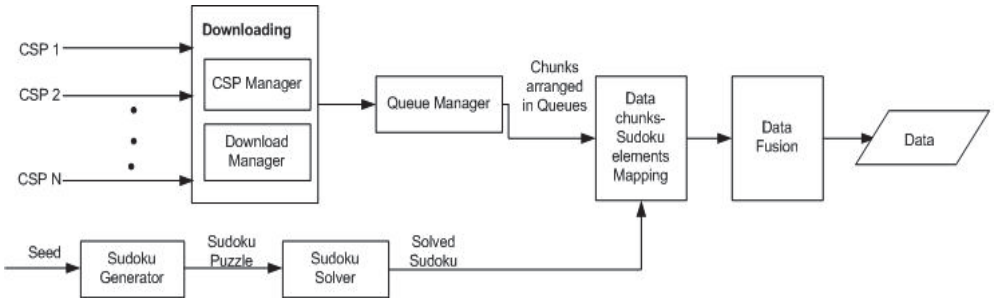

Fig. 2. Framework for downloading, unscrambling and fusion of data chunks.

Sudoku $[i, j]$ =

	1	2	3	4	5	6	7	8	9
1	2	9	5	7	4	3	8	6	1
2	4	3	1	8	6	5	9	2	7
3	8	7	6	1	9	2	5	4	3
4	3	8	7	4	5	9	2	1	6
5	6	1	2	3	8	7	4	9	5
6	5	4	9	2	1	6	7	3	8
7	7	6	3	5	2	4	1	8	9
8	9	2	8	6	7	1	3	5	4
9	1	5	4	9	3	8	6	7	2

Fig. 3. A sample solved Sudoku table.

placed in Sudoku [1, 1] to Sudoku [1, 9] and 10th chunk in Sudoku [2, 1] and this process continues until all the 81 chunks are placed. Here, the first data chunk will be moved to the queue (Enqueue) corresponds to the CSP ID-2, the second chunk will be moved to the queue corresponds to the CSP ID-9, and so on. The 17th data chunk, indexed as Sudoku [2, 8], will be the second entry to the queue corresponds to the CSP ID-2, and the 16th chunk, indexed as Sudoku [2, 7], will be the second entry to the queue corresponds to the CSP ID-9. Once all the elements are read and Enqueued, all the 9 queues correspond to 9 CSP IDs will be filled. Later, the Upload Manager will upload

the chunks in the queues into the corresponding CSPs, without making any changes in the order within the queues.

Algorithm 1: Split-Scramble-Upload-Store
Input: Data to be stored in CSPs
Output: Data chunks scrambled, and stored in N number of CSPs
1. Divide the data D of size M into Z (N*N) number of data chunks, of fixed size 'd'

$$D = d_1||d_2||d_3 \ldots||d_Z \text{ and } M = \sum_{i=1}^{Z} d_i$$

2. Generate an N*N Sudoku puzzle table
3. Find the solution for the puzzle
4. Select N number of CSPs and assign then identifiers (IDs) starting from 1 to N
5. Place the chunks into the solved Sudoku table
6. Store the data chunks into Queues associated for each CSP

```
k←1
for i = 1 to N do
   for j =1 to N do
        x ← Sudoku [i, j]
      Enqueue d_k to Q_x
   end for
k ← k+1
 end for
```

7. Store Q_x to CSP_x // The chunks, kept in each Queue, will be stored into the corresponding CSPS

The reconstruction of the original data in the private cloud is illustrated, in the following section, with the same example sudoku table shown in Fig. 3. All the data chunks stored in the N number of CSPs, in the form of queues, will be downloaded to the private cloud. Using the same seed value and the generator function, the same sudoku table will be created and solved. Thus, the same sudoku table, used for uploading and scrambling, will be created again. The first element in the sudoku table will be read, as earlier. This element is an index for the CSP ID and the queue in which the data chunk is placed. In this example, The data chunk is taken out from the queue and is logically placed in the location of the element in the table. In the example given in Fig. 3, the first element (Sudoku [1, 1]) is 2. Hence, the chunk in the front end of the queue, corresponding to CSP ID-2, will be taken out and placed in Sudoku [1, 1]. Next, Sudoku [1, 2] is read and the element is 9. Hence, the chunk in the front end of the queue, corresponding to CSP ID-9, will be taken out and placed in Sudoku [1, 2]. All the elements in the Sudoku table are read and similar dequeue and placement processes are performed. Eventually, the Sudoku table will be arranged with the data chunks in sequential order. The fusion of the chunks arranged in row-wise will return the original data.

Algorithm 2: Download-Unscramble-Fuse
Input: Scrambled data chunks stored in N number of CSPs

Output: Data

1. Generate the same Sudoku table used for uploading.
2. Download all the Queues from all the CSPs.
3. Re-arrange the chunks, from each queue

```
k ←1
 for i = 1 to N do
   for j =1 to N do                    //Reading the elements from each Queue
    x ← Sudoku[i, j];
        d_k ← Dequeue [Q_x];
   end for
    k ← k+1
 end for
```

4. $D = Join\ (d_i)^k$ //fuse chunks together in the original order to form the original data

4. PERFORMANCE EVALUATION

4.1 Security

The security of the proposed mechanism lies in the randomness achieved in data scrambling performed based on the sudoku puzzle, and the use of *N* number of CSPs for storing the private data.

Since the original data is split into multiple numbers of data chunks and stored into *N* number of CSPs, the probability of getting all the chunks together for an attacker is very less. The chunks stored in a CSP are not in sequential order; these chunks are scrambled and selected based on the sudoku puzzle and hence the information gained by an attacker who successfully hacked a CSP. The information existing or the uncertainty can be expressed in terms of Entropy,

$$E(n) = P(n) \times \log_2 (\frac{1}{P(n)}) \tag{1}$$

where $P(n)$ is the probability for 'n' events to occur.

The numbers in the range 1 to *N*, when arranged in Sudoku puzzle table, possess more Shannon Entropy than the numbers arranged in any other *N***N* matrices [21]. The number of classic 9×9 Sudoku solution grids is 6,670,903,752,021,072,936,960 or approximately 6.67×10^{21}. This is roughly 1.2×10^{-6} times the number of 9×9 Latin squares [22]. It shows that the randomness exist for selecting the chunks for each CSP is high and hence the information an attacker may infer, by hacking all the chunks stored in a CSP, will be very less.

The use of *N* number of CSPs to store various chunks increases the uncertainty and reduces the amount of information an attacker may gain. Since all the commercial CSPs might have possessed with reasonably good security mechanisms, hacking a CSP is not an easy task in the real scenario, but not impossible too. Even a CSP is compromised, the probability that all the chunks belonging to a particular data set are accessible to the at-

tacker is very less. Let Z be the number of nodes in a CSP, C be the number of chunks of a single data file stored in that CSP, and I be the number of nodes successful intruded, and if $I > C$, then the probability that I number of victim nodes contain all the data fragments represented by $P(I, C)$ is given as

$$P(I,C) = \frac{{}^{I}C_{C} \times {}^{(Z-I)}C_{(I-C)}}{{}^{Z}C_{I}} \tag{2}$$

(for *e.g.*, Z=100, I=20, C=10, $(I, C) = 0.000567$)

The above scenario shows how less is the probability that an attacker gains information about a data set if a CSP is compromised. The following two cases illustrate the less probability an attacker or a malicious CSP has to gain information about the data stored in N CSPs – (i) worst case – the attacker get access to all the correct N number of CSPs used for data storage in the first N attempts (ii) average case – the attacker get access to only 'y' number of CSPs, where $y > N$.

Worst case: The probability of getting all n number of CSPs in the first attempt of the attacker is given as:

$$P(N,G) = \left(\frac{N}{G}\right) \times \left(\frac{N-1}{G-1}\right) \times \left(\frac{N-2}{G-2}\right) \times \left(\frac{N-3}{G-3}\right) \ldots\ldots \left(\frac{N-(N-1)}{G-N}\right). \tag{3}$$

Where N is the selected number of CSPs for data storage and G is the available number of CSPs which provides storage as a service.

For *e.g.*, If N=9 and G=200, then the probability $P(N, G) = 8.5 \times 10^{-16}$

This is the worst case probability with respect to user and best case probability with respect to the attacker and the value is close to zero.

Then, the probability for an attacker gets all the fragments from a CSP, based on the above mentioned scenarios, can be given as

$$P = P(I, C) \times P(N, G). \tag{4}$$

For the given scenario, the probability value will be 4.81×10^{-19}.

Average Case Now consider that the attacker gets successful access in X number of CSPs among G number of available CSPs in the market and the secret data is stored in N number of CSPs. Then the probability that X contains all the N is given as

$$P(X,N) = \frac{{}^{X}C_{N} * {}^{(G-X)}C_{(X-N)}}{{}^{G}C_{X}} \text{ ; if } X>N. \tag{5}$$

For *e.g.*, if X=20, N=9, and G=200, then the $P(X, N) = 1.23 \times 10^{-5}$, that also a very less value.

As the number of CSPs N increases, the probability of successful attack will become very less in both worst and average case. So attacker needs more effort to retrieve private data from the large value of N.

4.2 Performance

A series of experiments are conducted to analysis the time required to execute SiDS and some keyed encryption algorithms, in a desktop machine which has Intel®i5™ CPU @ 2.5 GHz, 4GB RAM. The experiments were conducted separately for evaluating the time complexity of the SiDS (data splitting-then-scrambling and unscrambling-then-data fusion) and the time consumption for data splitting-then-uploading to CSPs and down-loading-then-data fusion. The execution time of the SiDS is compared with some block cipher algorithms like DES, Triple DES, and AES, which are standardized by NIST as well as very widely used for commercial purposes.

Table 1. Comparison of execution time of SiDS with block cipher algorithms, in ECB and CFC modes.

Data Chunk Size (in KB)	Execution Time in Seconds						
	ECB Mode				CBC Mode		
	DES	3DES	AES	*SiDS*	DES	3DES	AES
20	0.67	2.33	1.33	0.61	5.67	19.33	20.67
36	1.33	4.33	2	1.12	10	33	31.33
49	1.67	5.67	2.67	1.45	13.67	43.33	41.67
59	2.33	7.67	3.67	1.82	17.33	60.33	58
69	3	8.67	4.33	2.11	23	67	66.67
138	5.67	17	8.67	3.91	43	133.67	136.33
158	6.67	20	10	4.51	50.33	157.33	157.67
166	7	20.67	10.33	4.81	53	162.67	163
192	8	24	12	5.51	61.67	189.33	189
232	10	29	14.67	6.67	76.33	227	229

The time of execution of the proposed SiDS algorithm is compared with DES, Triple DES, and AES in ECB (Electronic Code Book) mode and CBC (Cipher Block Chaining) mode. The ECB mode is the simplest one; consume less time since it uses the key and the message as input for encryption. In CBS mode, other than the key, previously encrypted block is also used as input for encrypting new blocks of the messages. Though CBC consumes more time than ECB, CBC mode is commonly used since it provides much protection than ECB mode. The experimental results show the proposed SiDS consumes very less time compared to the block cipher algorithms even in ECB mode. The CBC mode is the commonly used and better mode in terms of security, and hence, the execution time of SiDS is compared with other block cipher algorithms in CBC mode also. The Table 1 shows the values obtained during the experiments conducted to compare the execution time of SiDS with the block cipher algorithms in EBC mode and CBC mode.

The Figs. 4 and 5 show the graphical representation of the values given in Table 1. The results show that in terms of execution time the SiDS outperforms the standard block cipher algorithms.

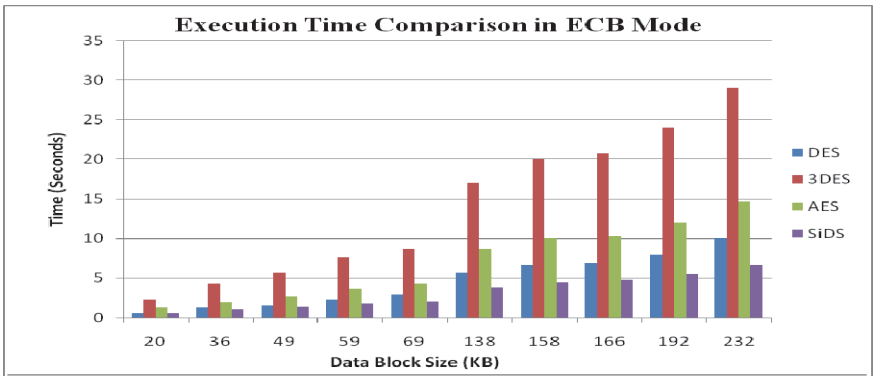

Fig. 4. Execution time results with ECB mode.

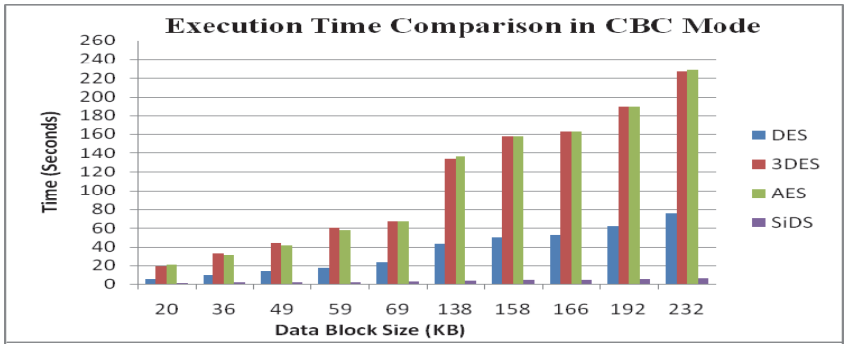

Fig. 5. Execution time results with CBC mode.

The values in Table 1 are the analysis of the time of execution of the algorithms without considering the time of uploading to and downloading from the CSPs. A separate analysis is done to evaluate the performance of splitting a data into multiple chunks and uploading to multiple CSPs, in parallel. The experimental results confirm that the SiDS consumes less time, to upload into and download from multiple CSPs than a security system that encrypts the whole data and uploads it into single CSP and download later for decryption. For this performance evaluation N is chosen as three, *i.e.* three CSPs are for storing the data chunks. The three CSPs used are Google Drive (G), Dropbox (D) and OneDrive (O) [23-25]. The experiments are conducted with two types of files : a small text file of size 22MB, and a large video file of size 933MB. Since the internet connection used was a shared type, the experiments were conducted for 5 different days and the upload and download speed were checked for 4 different times in a day, and the average values are considered. In the case of SiDS, the time is calculated for splitting the file, uploading the chunks into multiple CSPs, downloading the chunks from the CSPs and merging together. Here, the files are split into 9 or 21 chunks, and since only 3 CSPs are available, the chunks are arranged into 3 groups where each group consists of 3 or 7 chunks, and each group will be uploaded into each CSP. The uploading to and downloading from each CSP can be done in parallel. The results analyzed are given in Table 2.

Table 2. Time required to upload and download a file with and without data splitting and encryption mechanism.

File Type and Size	Time taken for each operations (Time in Seconds)				Total Time
Text File (22 MB)	Encryption-3	Upload-27	Download-14	Decryption-1	45
	Splitting (9 chunks) 1	Upload(3 chunks each)	Download(3 chunks each)	Merging 1	18
		G-8	G-7 7		
		D-8	D-7 7		
		O-9	O-7 7		
Video File (933 MB)	Encryption-40	Upload-350	Download-257	Decryption-35	682
	Splitting (21 chunks) 16	Upload (7 chunks each)	Download(7 chunks each)	Merging 13	358
		G-177 177	G-140 140		
		D-188	D-141		
		O-175	O-140		

5. CONCLUSIONS

The security and privacy issues existing in the cloud environment still prevent the widespread adoption of cloud. The loss of control over the data when it is stored in CSPs leads to security breaches. Even though the public CSPs holds good reputation in the industry, no CSPs won't ensure 100% security for their users' data. So, it is users' responsibility to keep data safe, when it is stored in the cloud as storage as service.

In this paper, *N* number of public CSPs are used to avoid the chance of single point of failure. Since the data is split into multiple data chunks and stored in multiple CSPs, even a CSP is compromised or even a CSP has become malicious, the chance of information leakage will be very less. Sudoku puzzle based scrambling provides a greater amount of randomness and hence, getting a meaningful information out of the chunks stored in a CSP is very difficult. In addition, the amount of information that needs to be stored in the private cloud side is very minimal and less informative. The strength of the security provided by the proposed work is analyzed with cryptographic algorithm metrics like randomness, uncertainty, and exposure probability. The time complexity and space complexity of the proposed work is less than the most commonly used cryptographic algorithms (DES and AES). This work has attempted to innovate an out of the box method for providing data security in cloud environment, which can be optimized further for better results. In future, by incorporating de-duplication concepts, time and space consumed by SiDS can be reduced.

REFERENCES

1. NIST, http://www.nist.gov/itl/cloud/, August 2015.

2. R. Venkateswaran, O. G. Basas, and R. T. Gutierrez Jr., "A survey on research issues in cloud computing," *International Journal of Sciences: Basic and Applied Research*, Vol. 16, 2014, pp. 449-454.
3. T. Jun, C. Yong, Q. Li, R. Kui, L. Jiangchuan, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," *ACM Computing Surveys*, Vol. 49, Article 13, 2016.
4. R. R. Rajavel and T. Mala, "Adaptive probabilistic behavioural learning system for the effective behavioural decision in cloud trading negotiation market," *Future Generation Computer Systems*, Vol. 58, 2016, pp. 29-41.
5. Hunter, "Cloud aloud [IT Applications]," *Engineering and Technology*, 2009, Vol. 4, 2009, pp. 54-56.
6. S. Gurumurthy, T. N. Babu, and G. S. Shankar, "An approach for security and privacy enhancing by making use of distinct clouds," *International Journal of Soft Computing and Engineering*, Vol. 4, 2014, pp. 214-217.
7. J. M. Bohli, N. Gruschka, M. Jensen, L. Iacono, and N. Marnau, "Security and privacy enhancing multicloud architectures," *IEEE Transactions on Dependable and Secure Computing*, Vol. 10, 2013, pp. 212-222.
8. K. Venkataramana and Padmavathamma, "Multi-tenant data storage security in cloud using data partition encryption technique," *International Journal of Scientific and Engineering Research*, Vol. 4, 2013, pp. 6-10.
9. S. J. Han, H. S. Oh, and J. Park, "The improved data encryption standard (DES) algorithm," in *Proceedings of IEEE 4th International Symposium on Spread Spectrum Techniques and Applications*, Vol. 3, 1996, pp. 1310-1314.
10. P. Mahajan and A. Sachdeva, "A study of encryption algorithms AES, DES and RSA for security," *Global Journal of Computer Science and Technology Network, Web & Security*, Vol. 13, 2013, pp. 15-22.
11. B. Balamurugan and P. V. Krishna, "An enhanced security framework for a cloud application," *Artificial Intelligence and Evolutionary Algorithms in Engineering Systems*, 2015, pp. 825-836.
12. H. Hacigumus, B. Iyer, and S. Mehorotra, "Providing database as a service," in *Proceedings of the 18th IEEE International Conference on Data Engineering*, 2002, pp. 29-38.
13. S. Bajpai and P. Srivastava, "A fully homomorphic encryption implementation on cloud computing," *International Journal of Information and Computation Technology*, Vol. 4, 2014, pp. 811-816.
14. I. Ahmad and A. Khandekar, "Homomorphic encryption method applied to cloud computing," *International Journal of Information and Computation Technology*, Vol. 4, 2014, pp. 1519-1530.
15. K. S. Vijayanand and T. Mala, "Securing data in multicloud hybrid model using data splitting and encryption," *International Journal of Applied Engineering Research*, Vol. 10, 2015, pp. 75-80.
16. R. A. Bailey, P. J. Cameron, and R. Connelly, "Sudoku, gerechte designs, resolutions, affine space, spreads, reguli, and Hamming codes," *American Mathematical Monthly*, http://www.maths.qmul.ac.uk/~pjc/preprints/sudoku.pdf.
17. J. Gunther and T. Moon, "Entropy minimization for solving Sudoku," *IEEE Transactions on Signal Processing*, Vol. 60, 2012, pp. 508-513.

18. L. Arson, "Sudoku science," *IEEE Spectrum*, Vol. 43, 2016, pp. 16-17.
19. Mathematics of Sudoku, https://en.wikipedia.org/wiki/Mathematics_of_Sudoku, February, 2016.
20. Sudoku, https://en.wikipedia.org/wiki/Sudoku, February, 2016.
21. K. N. Paul and A. D. Stephen, "The Shannon entropy of Sudoku matrices," in *Proceedings of the Royal Society A*, No. 466, 2010, pp. 1957-1975.
22. J. Frazer, "Sudoku enumeration problems," Frazer Jarvis's home page, February 6, 2016.
23. Google Drive, http://en.wikipedia.org/wiki/Google_Drive, 2015.
24. Dropbox, http://en.wikipedia.org/wiki/Dropbox, 2015.
25. OneDive, http://en.wikipedia.org/wiki/OneDrive, September 2015.

K. S. Vijayanand, Assistant Professor in Information Technology, Government Engineering College, Barton Hill, Kerala, India, received his Bachelor's degree in Information Technology from Cochin University of Science and Technology in the year 2002 and Master's degree in Computer Science and Engineering from Anna University in the year 2010. He is currently pursuing Ph.D. in the Department of Information Science and Technology, Anna University, Chennai, India. His research interest includes trust and data security in cloud services. He is a Lifetime member of Indian Society for Technical Education.

Mala Thangarathinam is an Associate Professor in the Department of Information Science and Technology, Anna University, India. She completed her Ph.D. on NLP at Anna University during the year 2008. Currently, she is guiding more than ten research scholars. Her research areas include natural language processing, virtualization technologies, grid computing and cloud computing. She has more than 10 research publications in national conferences, 20 research publications in international conferences and 12 research publications in international journals.