DNA Based Cloud Storage Security Framework Using Fuzzy Decision Making Technique

Abhishek Majumdar[1*], Arpita Biswas[1], Krishna Lal Baishnab[1] and Sandeep K. Sood[2]
[1] National Institute of Technology Silchar,
Assam, India
[e-mail: abhishekmajumdar91@gmail.com]
[2] Guru Nanak Dev University, Regional Campus,
Gurdaspur, Punjab, India
[e-mail: san1198@gmail.com]
*Corresponding author: Abhishek Majumdar

Received April 19, 2017; revised November 21, 2018; revised January 3, 2019; accepted January 24, 2019; published July 31, 2019

Abstract

In recent years, a cloud environment with the ability to detect illegal behaviours along with a secured data storage capability is much needed. This study presents a cloud storage framework, wherein a 128-bit encryption key has been generated by combining deoxyribonucleic acid (DNA) cryptography and the Hill Cipher algorithm to make the framework unbreakable and ensure a better and secured distributed cloud storage environment. Moreover, the study proposes a DNA-based encryption technique, followed by a 256-bit secure socket layer (SSL) to secure data storage. The 256-bit SSL provides secured connections during data transmission. The data herein are classified based on different qualitative security parameters obtained using a specialized fuzzy-based classification technique. The model also has an additional advantage of being able to decide on selecting suitable storage servers from an existing pool of storage servers. A fuzzy-based technique for order of preference by similarity to ideal solution (TOPSIS) multi-criteria decision-making (MCDM) model has been employed for this, which can decide on the set of suitable storage servers on which the data must be stored and results in a reduction in execution time by keeping up the level of security to an improved grade.

Keywords: Cloud Storage Security, DNA Cryptography, Fuzzy Logic, MCDM

1. Introduction

The current commercial leaders in personal cloud storage expected that each cloud user will store 3.3 Tb of their personal data in cloud in the very near future [1]. Despite the cloud technology achieving much popularity, most organizations or users are threatened to adopt the cloud technology only because of the paucity of securities in it. The cloud may experience the ill effects of several vulnerabilities because of the plan, programming or configuration mistakes of designers and service providers at different architectural layers, which are infrastructure, platform and application that can trade off the evaluation of the contracted quality of service (QoS). Moreover, the cloud also turns out to be the favourite target for attackers, where they can practice obnoxious activities. Therefore, an appropriate and more protective cloud security is vital in projecting the big cloud market. Even the International Data Corporation's (IDC) findings show that security concerns are the most crucial issue that cloud computing is facing [2]. Data may have to be encrypted by data owners before outsourcing to the commercial public cloud to secure highly sensitive data and restrict unauthorised accesses in the cloud and beyond [3]. As a result, the traditional plaintext keyword search-based data utilization service would not be possible. Moreover, downloading all the data and decrypting them locally would become more impractical because of the tremendous measure of the data transfer capacity cost in cloud scale frameworks. Thus, exploring an effective search service and preserving privacy over encoded cloud information are of foremost significance. However, the encrypted cloud data search framework remains a tough assignment for today's distributed computing systems because of different inalienable security and protection boundaries, including different strict prerequisites like the index privacy, data protection, keyword privacy and numerous others [4, 5].

Deoxyribonucleic acid (DNA)-based cryptography is taken as the most promising area of security by several researchers because of its complex structural features and several interesting characteristics. Few researchers adopted DNA computing, while others used the biological properties of DNA in their approaches for information security. However, all these schemes possess many drawbacks in high data accessing time, high data searching time, high system overhead and various security issues. This paper proposes a novel DNA-based cloud storage framework to solve the aforementioned issues of the existing approaches. The major contributions of this paper are as follows:

a. A DNA variant of Hill cipher encryption scheme is followed for a strong 128-bit key generation.
b. Many problems, including high data accessing time and high searching time of data, are minimised by introducing a DNA-based data indexing method.
c. The data in the proposed scheme have been classified based on different qualitative security parameters obtained using a specialised fuzzy-based classification technique.
d. Meanwhile, a fuzzy-based technique for order of preference by similarity to ideal solution (TOPSIS) multi-criteria decision-making model is designed to reduce the execution time by keeping up the level of security to an improved grade. This model can decide on the set of suitable storage servers, on which the data must be stored.

In addition, security, sensitivity and functionality analyses are presented in this paper. These analyses prove that the proposed scheme is more efficient and secured than the existing schemes.

The rest of the paper is organised as follows: Section 2 presents the related works in the relevant field in detail. Section 3 addresses the background of this area. The proposed cloud storage architecture is presented in Section 4. Section 5 presents results and different kinds of analysis. The paper is concluded in Section 6.

2. Related Work

Many researchers proposed various cloud storage and security frameworks for the past decades. However, enormous research for enhancement is still on-going because of the unabatedly increasing number of cloud users and security issues.

Research on cryptography-based cloud storage was first reported in the work of Kamara et al. [3]. They developed secure cloud storage architectures for both consumer and enterprise scenarios by exploiting the features of non-standard cryptographic techniques, such as attribute-based encryption and searchable encryption. Their work highly focused on attaining confidentiality, which was rectified on their later works by introducing other properties like integrity, searchability and verifiability [6]. Wang et al. [7] emulated the data security problem in cloud data storage, which is a distributed storage system. Their work proposed an efficient scheme with an exceptional data support and having added features, such as block update, delete and append. The erasure-correcting code technique is implemented herein to prepare the file distribution system, keeping in mind a guaranteed data dependability. The method involves an integration of data error localization and storage correctness insurance. Their scheme is also robust, has an inordinate efficiency and is resilient to various failures and attacks, such as Byzantine failure. The authors also present a refined verification scheme public auditing system with the protocol that supports complete dynamic data operations [8]. Accordingly, the existent proofread of the PDP or PoR scheme was improved by spoofing the basic Markle Hash Tree. This approach was capable of performing many auditing jobs with the help of an efficient third-party auditing (TPA) using a bilinear aggregate signature technique. However, the computation cost of the Boneh–Lynn–Shacham (BLS) algorithm used during the TPA was a big issue here. Moreover, their model was unable to support both public verification and dynamic data correctness. The computation power in the cloud computing environment is provided by an assembly of storage servers that are usually installed with the help of hundreds to thousands of servers [9]. The authors herein modelled a four-layered typical Cloud-based data. The lowest layer was made of tremendous physical resources (i.e. storage and application servers) that helped in powering the storage servers. The servers straightforwardly oversaw the higher-level virtualization services and toolkits that permitted sharing of their ability among virtual instances of servers. However, the virtual occurrences were secluded from each other, leading to an isolated security context, resulting in a fault-tolerant behaviour [10]. As regards security purposes, cryptography has always been an essential part of cloud computing. Cloud computing is mostly based on replication, and in this manner, keeping up the uniqueness of the encryption and decryption keys is vital. The cloud systems of Amazon have recently been confronted with a challenge regarding this issue. However, the absence of prescience in cryptography can prompt deplorable results [11]. This application involves extensive concentration on the encryption and key distribution. A comprehensive security framework for cloud computing environments was proposed by Takabi et al. [12]. Their model described some approaches that deal with security challenges. The model was composed of different modules to handle security. The modules addressed issues, such as identity management, access control, policy integration among multiple clouds and trust management between different clouds and between a cloud and its users.

Venkatesan et al. [13] proposed a proficient multi-agent-based static and dynamic data integrity protection by periodically confirming the hash estimation of the files stored in the massive data storage. Their proposed model depended on the multi-agent system (MAS). The agent here had a capacity of self-ruling, ingenuity, social ability and so on. The proposed architecture incorporates three entities (i.e. client, service provider and data owner) and has different agents to screen and keep up the data integrity. Different security aspects in computing were discussed by Prasad et al. [14] and Sood et al. [15]. Techniques involving a method of dealing with verifying in three-dimensional (3D) strategies were projected by Prasad et al. [14] with a noteworthy detriment, wherein the data stored are not in encrypted form. However, the 3D approach provides the data availability by overcoming problems, such as data leakage and denial of services. In addition, it is more flexible for a complex and diverse network. Sood et al. [16] proposed a framework that consisted of different procedures and techniques to protect data. They introduced the concept of data security sections, which is followed in our paper in a different manner. Confidentiality, availability and integrity parameters for cryptography in addition to the message authentication code (MAC) for checking the data integrity are utilised as part of this procedure. The strategy provides classification, uprightness, authorization, verification and non-repudiation and anticipates data spillage. The security degree that they provide in ascending order is MAC, classification of data and execution of index and encryption system. Researchers like Spillner et al. [17] contributed to storage optimality for storage service and controller architecture for storage, which is conscious of optimality, by proposing a new prototype, called NubiSave. However, NubiSave needs to integrate with frontends to achieve a critical mass of actual users. Dong et al. [18] presented the process of efficient access and storing of small files with storage on a Hadoop distributed file system. The prefetching technique was used to obtain a better access efficiency. The cut-off point was measured to improve the I/O performance, but the formula to measure the cut-off point was not defined in their work. Xia et al. [19] proposed "Greedy Depth first Search" algorithm to establish a multi-keyword searching scheme with the dynamic updating capability. Furthermore, a tree based index structure was also proposed using the KNN algorithm. Cui et al. [20] presented a secure attribute-based cloud storage framework that supports secure data provenance, fine-grained access control, dynamic user management and other security assurances.

Many researchers nowadays are incorporating the DNA concept to improve the data security. Abbasy et al. [21] proposed an approach to camouflage confidential data using DNA reference sequences. This approach consisted of an initial data conversion from binary to DNA nucleotide sequences for encryption purposes. The process is followed by the application of complementary rules on encrypted data and the determination of the index of each couple of nucleotides in a DNA reference sequence, wherein the encrypted data are stored. The bottom–up traversal of the same approach was followed for decryption purposes. This approach provided security and privacy of the user's data in the resource-sharing environment. Khalifa et al. [22] proposed a steganographic scheme of text hiding, wherein the text is transformed to a collection of amino acids, then encrypted using a DNA-based playfair cipher. A two-by-two complementary rule is then applied to hide the resultant cipher text in a DNA sequence. Atito et al. [23] showed the combination of DNA-based cryptography and the data-hiding technique. The scheme was implemented in two consecutive stages as follows: the first stage dealt with the encryption of plain text and amino acid-based playfair cipher; and the second stage dealt with the insertion method, where a reference DNA sequence was used to hide the encrypted DNA cipher text. Pramanik et al. [24] proposed a cryptographic scheme using DNA hybridization and DNA digital coding using OTP that utilised a single-stranded

DNA string as the secret key. The plain text was encrypted using a DNA key, whose length depended on the plain text itself. In addition, the plain text was fragmented into several DNA sequence packets during transmission by attaching the packet sequence number with it and sending it to the receiver one by one. Kar et al. [25] presented a DNA-based round encryption method, where two encryption methods were followed. The first round was based on the cipher block chaining method, where a random number generator was used to generate the round one key. In the second round, a DNA sequence was selected from the pool of sequences and segmented in 8-bit blocks. Addition was then performed between the two rounds' texts, and the resultant was converted into a fake DNA sequence using a binary coding scheme and the DNA primer concept. However, this approach required extra space, which became its limitation. Sadeg et al. [26] proposed a symmetric key block cipher inspired by the biological characteristics of DNA and the protein synthesis mechanism.In their work, a series of substitutions and permutations were performed to improve enforce the security in their approach. Wang et al. [27] proposed a new technique to show how cryptography works with DNA computing. In their approach, the combination of DNA computing and the RSA algorithm was made to efficiently encrypt a message. Meanwhile, Gehani et al. [28] introduced the concept of one-time-pads in the DNA cryptography. The encryption using a one-time-pad was based on randomness and has the advantage of having an unbreakable and unpredictable nature. As a result, each encryption was unique and did not bear any relation to the next encryption. The author also described the use of a substitution approach with random paired mapping and exclusive OR scheme with nucleotide computation in DNA cryptography. Marwan et al. [29] proposed a modified DNA-based playfair cipher encryption technique and performed a comparative analysis of various combined techniques in terms of security, speed, data size and hiding capacity.

3. Background Study

Deoxyribonucleic acid (DNA) is a molecule that acts as a carrier of the biological instructions an organism needs to develop, live and reproduce [30]. Each cell in the human body contains a nucleus, which characterizes all the physical and behavioral features of the human body. They are packed into chromosomes. A DNA is shaped like a double helix structure made up of two strands, where each strand can consist either a purine or a pyrimidine base. The purine bases are adenine (A) and guanine (G), while the pyrimidine bases are thymine (T) and cytosine (C), also known as the four basic nucleotide bases of DNA. The two strands in a double helix DNA are linked together. The bases are bonded with each other by hydrogen bonds (i.e. A with T and C with G) and are called the complementary pairs of DNA strands. Every three adjacent nucleotide bases in a DNA sequence form a codon that maps to a unique amino acid used in protein synthesis. Another concept of DNA is the primer, which is nothing but a DNA sequence appended on both sides of the original DNA sequence that makes it difficult for the intruder to identify the original DNA sequence. However, these natural rules and structures are used in cryptography in different ways for security concerns and to increase the information complexity.

4. Proposed Model

An efficient method is proposed here to provide better security and reliable data storage and transmission. **Fig. 1** shows the proposed framework for a secure cloud storage. The proposed approach manages the techniques to store the owner's data in an encrypted form into several

cloud storage servers registered with the cloud service provider (CSP). The proposed method achieves a secured environment through several modules: DNA–Hill cipher key generation, DNA-based index building and encryption, fuzzy-based security classification and fuzzy-based storage server selection and information storing.

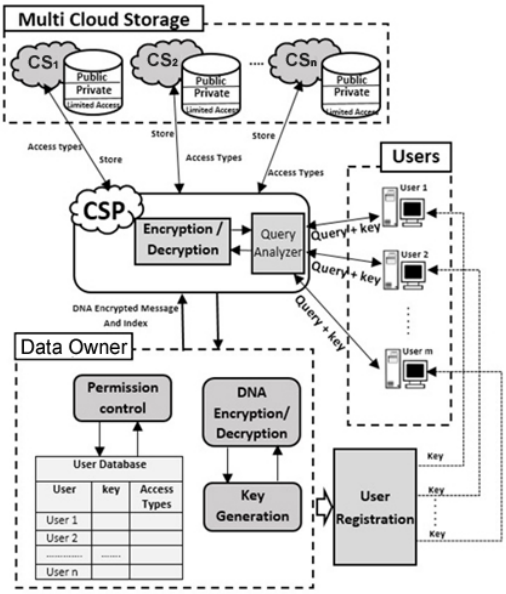


Fig. 1. Proposed Cloud Data Security Model

4.1 Key Generation

The very first phase of the proposed model is the key generation. In this work, a proposed DNA-based hill cipher strategy is followed for key generation. This strategy is inspired by the traditional Hill cipher symmetric encryption due to its several advantages such as masquerading letter frequencies of the plaintext and high throughput. Even though the traditional hill cipher has speed and ease, it becomes obsolete due to its vulnerability nature against known plaintext attack [31]. Thus, for enhancing the security of Hill cipher, the DNA-variant of this algorithm is proposed. The proposed DNA-based hill cipher strategy for key generation is depicted in Fig.2. In this phase, firstly, a 64 nucleotides long DNA string K_{DNA} is chosen randomly. Along with this, another 16- nucleotides long DNA sequence S_D is also made by considering every 4^{th} DNA base present in K_{DNA}. Subsequently, the S_D is split into four equal parts with each part of S_{Di} consisting combinations of four DNA bases. As per the Table 1, each S_{Di} is then mapped to its corresponding value N_i. As listed in **Table 1**, a total of 251 values (from 0 to 250) are taken for the possible 256 combinations of a four-character DNA strand. Each value from 246 to 250 signifies two ambiguous DNA strands. The five ambiguous values may also be set with other combinations as per the data owner's wish to make it complex and make tougher for the attacker guess.

The values of each N_i are then arranged as a 2×2 invertible, modulo 251 matrix K_{DH}, whose elements range from 0 to 250. Mathematically, $(p^2 - 1) * (p^2 - p)$ invertible 2×2 matrices can be found in modulo p, in which each entry can be from the set $\{0, 1, 2.... (p - 1)\}$, where p is prime. Thereafter, the proposed DNA–based key generation method is performed between the key matrix S_{DH} with every part of the DNA string K_{DNA} and the resultant DNA string K_{Ei} is computed. The whole key generation process is done as per Algorithm 1. Ultimately, the final

DNA encrypted 128 bit key K_{fin} is produced by combining all the K_{Ei}. This key will be used in the message encryption phase later on.

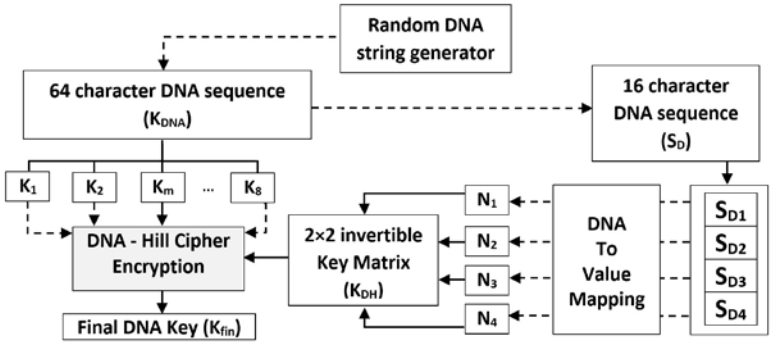


Fig. 2. DNA-Hill cipher-based key generation

Table 1. DNA strand to value mapping

Value	DNA string	Value	DNA string	Value	DNA string	Value	DNA string
0	AAAA	64	TAAA	128	CAAA	192	GAAA
1	AAAT	65	TAAT	129	CAAT	193	GAAT
2	AAAC	66	TAAC	130	CAAC	194	GAAC
3	AAAG	67	TAAG	131	CAAG	195	GAAG
..	……..	..	……..	..	……..	..	……..
..	……..	..	……..	..	……..	245	GGTT
59	AGCG	123	TGCG	187	CGCG	246	GGTC/ GGCG
60	AGGA	124	TGGA	188	CGGA	247	GGTG/ GGGA
61	AGGT	125	TGGT	189	CGGT	248	GGCA/ GGGT
62	AGGC	126	TGGC	190	CGGC	249	GGCT/ GGGC
63	AGGG	127	TGGG	191	CGGG	250	GGCC/ GGGG

Algorithm 1. Proposed DNA-Hill Cipher Algorithm for key generation

1. Select 64 nucleotide (128-bit as per **Table 1**) long DNA sequence K_{DNA} and split into 8 equal parts K_1, K_2… K_8 of 16 bit each.
2. Each K_i is further split into two 8 bit parts and taken as a column vector, padding as necessary.
3. Consider every 4th DNA base in K_{DNA} and generate a 16 nucleotide (32 bit) long DNA sequence, S_D.
4. Split S_D into four equal parts of four DNA characters each, S_{D1}, S_{D2}, S_{D3} and S_{D4}.
5. Map each S_{Di} into their corresponding values as per **Table 1**: N_1, N_2, N_3, and N_4.
6. Produce a 2 × 2 invertible key matrix 'K_{DH}' containing N_1, N_2, N_3, and N_4.
7. Encryption:
   Then cipher text block K_{Ei} is given by
   $K_{Ei} = K_{DH}\, K_i \pmod{251}$
   Produce Final Key $K_{fin} = K_{E1}|K_{E2}| \ldots .. |K_{E8}$

4.2 Data encryption and index building

Strong data encryption and searchable index building techniques will then be very necessary after a successful key generation. The index will help to search over the encrypted data. This study proposes a DNA-based encryption technique, where the owner's data will be

transformed into a sequence of DNA bases. A DNA strand only has four nucleotides (i.e. adenine (A), thymine (T), cytosine (C) and guanine (G)). Thus, the index IM′ will be more efficient and searchable because any keyword has to be searched among the combinations of those four nucleotides only. The keywords here are the strands of interest, which are nothing but the DNA encrypted word that a user might need to search later. The index will contain the list of DNA strands with an outline of pointers to the documents, where the strand occurs. As a result, a faster retrieval of the file would be possible. For each DNA strand of interest, say 'ATAC', a conceivable approach for index development would be made to list all the documents that contain DNA strand ATAC. A secret key that allows only a legitimate access to be able to decode the encrypted information.

Fig. 3 shows the proposed DNA encryption approach. In the encryption phase, for every 256-bit part of plain text the data are encrypted using a DNA-based encryption scheme, where the data at the owner side are first converted into their equivalent DNA sequence termed as M^{DNA} as per the inverse mapping of **Table 1**. The DNA complementary rules are then applied in the resultant sequence of the DNA strands. The complementary rules are generally used to puzzle the attackers over the ordering of the DNA bases. Generally, it should be in the format of $x \neq c(x) \neq c(c(x)) \neq c(c(c(x)))$ and $x = c(c(c(c(x))))$ [22, 23]. The complementary rules (A to T), (T to C), (C to G) and (G to A) are used in the proposed scheme, which implies C (X) = Y and so on. These rules may be changed every time for encryption as per the data owner's desire. After applying the complementary rules, the DNA sequence is now changed to M^{DNA_C}.

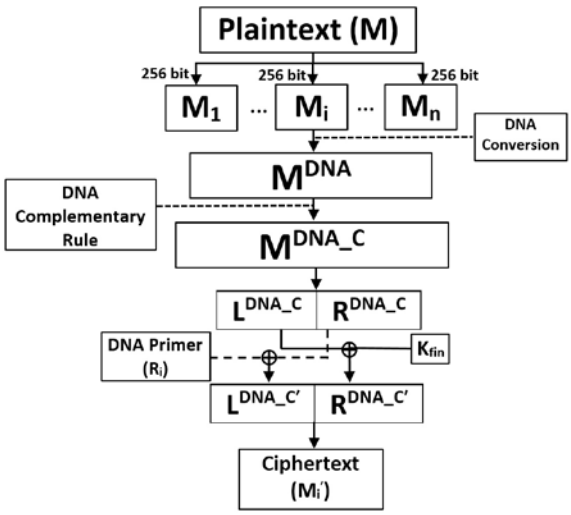


Fig. 3. Principle of the proposed encryption scheme

Finally, the cipher text will be constructed as follows: (1) the left part of the cipher text will be set as $L^{DNA\_C'}$, which is the result of the XOR function of the R^{DNA_C} with the 128 bit generated DNA primer sequence R_i; (2) the right part of the cipher text will be set as $R^{DNA\_C'}$, which is the result of the XOR function of the L^{DNA_C} with the 128 bit computed key K_{fin}; and (3) the cipher text (M_i^{CT}) will be formed by amalgamating $L^{DNA\_C'}$ and $R^{DNA\_C'}$. Lastly, all the resultant blocks for each 256 bit plaintext block will be combined together to form the final encrypted ciphertext M′. Here, each primer is of 128 bit in size and generated by amalgamating various pieces of information such that time and date, length of the original plaintext, length of extra padded bits, data owner id and access type. It will be used to check the integrity of the message at the receiver side later. **Fig. 4** represents the information present inside a primer with their

corresponding sizes in bits. After DNA encoding, the 256-bit SSL is used for establishing a secure transmission network. In this phase, the encrypted message M′ is again encrypted with the 256-bit SSL encryption having K_1 as the 256-bit secret symmetric session key that results into M″, which will be sent to the CSP for storing. After the SSL encryption, it becomes much terrible for the hackers or intruders to crack the code through a brute force attack. The encrypted data M″ along with index IM′ will be sent to the CSP for storage.

$$\underbrace{Primer\ (R_i)}_{128\ bit} = \underbrace{(sec}_{8\ bit}|\underbrace{min}_{8\ bit}|\underbrace{hh}_{8\ bit}|\underbrace{dd}_{8\ bit}|\underbrace{mm}_{8\ bit}|\underbrace{yy}_{8\ bit}|\underbrace{length(PT)}_{48\ bit}|\underbrace{P\text{-}bit}_{16\ bit}|\underbrace{DO_{id}}_{16\ bit}$$

Fig. 4. Structure of primer

4.3 Fuzzy based Security classification

This phase comes after the encryption phase at the data owner side. After a successful login of the data owner to the CSP, the data owner will store the data as per his desired security criteria. Storing is done by listing the desired P, Q and R security parameters, where P stands for proactive threat detection and management; Q stands for quality data backup; and R is the maximum uptime and minimum downtime, and sending them all together to the CSP. The encrypted message M′, encrypted index IM′ of the frequently searched keywords by the user and the secret key K_1 discussed in the earlier section are also sent along those three parameters.

A fuzzy-based approach is presented herein to store data as different access types in different cloud storage servers depending upon the three-abovementioned important security parameters (i.e. P, Q and R). The proposed fuzzy based approach has been used to classify the access types where uncertainty or fuzziness will be resolved using membership functions with a less execution time and without an extra burden of dataset training. The CSP provides different fuzzy variables for each security parameter as a choice to the user. The user itself will choose these variables. A security factor (S_f) will be figured from these selections by utilizing the proposed strategy shown in the sections that follow.

The abovementioned security parameters are qualitative in nature, and the user may not be aware of the procedure to give values to those parameters. Hence, the fuzzy linguistic variables will help the user to list their necessity in a finer level. For example, the CSP employs a linguistic variable rating range S = {VP, B. VP & P, P, B.P & F, F, B.F. & G, G, B. G & VG, VG}, where VP = Very Poor, B. VP & P = Between Very Poor and Poor, P = Poor, B. P & F = Between Poor and Fair, F = Fair, B. F & G= Between Fair and Good, G=Good, B. G & VG = Between Good and Very Good and VG= Very Good, to represent the three parameters and evaluate their importance.

The membership function for the triangular fuzzy is used to convert the linguistic variables into their corresponding linguistic scales. The membership function for the triangular fuzzy can be defined as follows using Eq. (1):

$$\mu_{\tilde{a}}(x) = \begin{cases} 0, & x < a_1 \\ \frac{x-a_1}{a_2-a_1}, & a_1 \le x \le a_2 \\ \frac{x-a_3}{a_2-a_3}, & a_2 \le x \le a_3 \\ 0, & x > a_3 \end{cases} \quad (1)$$

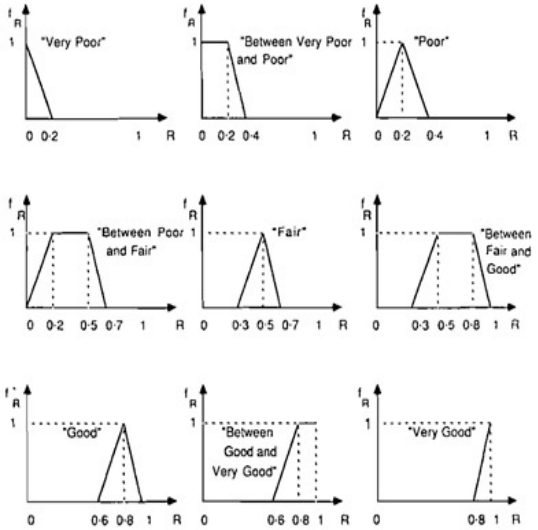


Fig. 5. Membership functions of the linguistic variables

Table 2 shows the linguistic scales of the variables mentioned in **Fig. 5**. For example, a user chooses the linguistic variable 'F' for parameter availability, 'B. G & VG 'for confidentiality and' B.F. & G' for integrity. The CSP then checks that all the parameters corresponding to the fuzzy linguistic terms chosen by the user are F= [0 0.3 0.5 0.7]; B. G & VG= [0.6 0.8 1 1]; B.F. & G = [0.3 0.5 0.8 1]. **Table 3** depicts an example of the access type calculation using a S_f value with some specified ranges. The graded mean representation and centroid methods are followed to find the S_f value from the fuzzified value of T_i.

Table 2. Linguistic scale to evaluate P, Q, and R

Linguistic variables	Linguistic ratings
VP	[0 0 0 0.2]
B. VP & P	[0 0.2 0.2 0.4]
P	[0 0 0.2 0.4]
B.P & F	[0 0.2 0.5 0.7]
F	[0 0.3 0.5 0.7]
B.F. & G	[0.3 0.5 0.8 1]
G	[0.6 0.8 1 1]
B. G & VG	[0.6 0.8 1 1]
VG	[0.8 1 1 1]

Table 3. Ranges of S_f for the access type calculation

a) Ranges of S_f for data access types						
Access Type	**Range of S_f**					**Access Type**
Public	$0 \le S_f \le 0.399$					Public
Private	$0.4 \le S_f \le 0.799$					Private
Owners Limited access	$0.8 \le S_f \le 1$					Owners Limited access
b) An example of access type calculation at CSP end						
User	**P**	**Q**	**R**	**$T_i=\sum$ [PQR]**	**S_f**	**Access type(A_T)**
U_1	[0.3 0.5 0.5 0.7]	[0.6 0.8 1 1]	[0.3 0.5 0.8 1]	[1.2 1.8 2.3 2.7]	0.6421	Private

U_2	[0 0.2 0.5 0.7]	[0 0 0.2 0.4]	[0 0.2 0.2 0.4]	[0 0.4 0.9 1.5]	0.2267	Public
U_3	[0.6 0.8 0.8 1]	[0.6 0.8 1 1]	[0.8 1 1 1]	[2 2.6 2.8 3]	0.7439	Private

4.4 Storage server selection and data storing

The data will be transmitted to the cloud for storage after data encryption at the owner side. In the proposed approach, the data will be stored in different independent and geographically distributed storage servers, instead of storing at a single server. This phase is used to select the available storage servers registered with the CSP for data storage. Some cloud service providers receive the data from the data owner and fragment them in several parts. Each data part is then stored on different geographical locations by selecting different storage servers with different storage types, level of security, etc. The task of data storage is being distributed among different access level locations (depending on S_f) on different storage servers to have an efficient, secure and faster computation.

4.4.1 Calculation of the weights of criteria

Data security is the most concerning area. Hence, the level of security of the storage server is the most important factor among all. The processing speed and the time delay are considered as the two most significant parameters in storing data in parallel on multiple storage servers because they save execution time and network bandwidth, thereby helping to decide on the extra communication cost in a cloud environment. However, the need for these criteria can be distinctive in different circumstances.

Here, the weights of the criteria is generated by pairwise comparisons for each of the chosen criteria using Analytic Hierarchy Process (AHP). AHP transforms the comparisons, which are most often empirical, into numerical values for further processing and comparison. The relative significance scale between two criteria as suggested by Saaty [32] is the most widely used. **Table 4** shows the scale where values vary from 1 to 9, determines the relative importance of a criterion when compared with another criterion. The consistency ratio (CR) measures the uniformity of a respondent's answers to the AHP questionnaires. The pairwise comparison matrix is now generated as shown in **Table 5(a)** by utilizing the Saaty's significance scale [32] listed in **Table 4**. **Table 5(b)** lists the final weight of each criterion calculated using AHP method. These weights will be used in the TOPSIS method later on for further processing in storage server selection. In this work, the obtained consistency rate is 0.0565 < 0.1~10%. Since the value is less than 10%, the weights can be considered consistent and can be used in further decision-making process.

Table 4. Significance scale of criteria [32]

Definition	Intensity of significance
Equally important	1
Moderately more important	3
Strongly more important	5
Very strongly more important	7
Extremely more important	9
Intermediate	2,4,6,8

Table 5. Weight determination of criteria

a) Pairwise comparison matrix for the criteria

Criteria	No. of CPU	Avg. Processing speed	Level of security	Avg. Transmission speed	Avg. Time delay	Avg. Memory utilisation
No. of CPU	1	1/3	1/7	1//5	1/3	3
Avg. Processing speed	3	1	1/3	1/3	2	6
Level of security	7	3	1	3	5	8
Avg. Transmission speed	5	3	1/3	1	4	7
Avg. Time delay	3	1/2	1/5	1/4	1	6
Avg. Memory Utilisation	1/3	1/6	1/8	1/7	1/6	1

b) Result obtained from the AHP

Criteria	Weights (W)	λ_{max}, CI,RI	CR
No. of CPU	0.0509716	Max. Eigen value λ_{max}= 6.37345 CI= 0.0746909 RI= 1.32	0.0565
Avg. Processing speed	0.138474		
Level of security	0.422752		
Avg. Transmission speed	0.260775		
Avg. Time delay	0.0997117		
Avg. Memory Utilisation	0.0273168		

4.4.2 Selection of Multiple Storage servers

Fuzzy set theory combined with multi-criteria decision making methods can be used to deal with the uncertainty in the data storage center selection process. Several criteria must be considered in the selection process, both quantitative and qualitative. Hence, a combination of these two techniques is adopted here to deal with the storage server selection problem.

A fuzzy-based TOPSIS approach is employed in the proposed model to select the best possible storage servers under real-time environments, where the ratings of various storage servers under various criteria are assessed in linguistic terms represented by fuzzy numbers. **Fig. 6(a)-(c)** depict the fuzzy set for the three qualitative criteria (i.e. level of security, memory utilisation and time delay), respectively, whose linguistic scales are shown in **Table 6**. **Table 7** lists some important criteria for the storage server selection.

A one-to-one correspondence of each storage server registered with the CSP with their corresponding attributes exists. Some of them are static, while some are taken in real time. The values of the qualitative terms, such as the level of security, average memory utilisation and average time delay are considered in **Table 6**. The quantitative attribute values are simply computed from the different storage server's current behavior, whereas the qualitative attributes are expressed in fuzzy linguistic terms. Ranking the storage servers as per their selection priority becomes possible by following Algorithm 2.

After obtaining the above list, the required number of globally distributed storage servers will be taken as per the data size and the section type to store the owner's data parts separately. The best storage servers computed at that time will be chosen to store different parts of the data on them if the size of the data is big enough and confidential (based on S_f). In addition, the CSP

generates an index IM″ by updating the encrypted index IM′ with another column containing pointers to locate the storage servers, in which each keyword is stored. The CSP creates a secure channel with them before storing the owner's data on different security sections on different storage servers and sending the data along with the owner's ID, access section and the CSP_ID for security concerns.

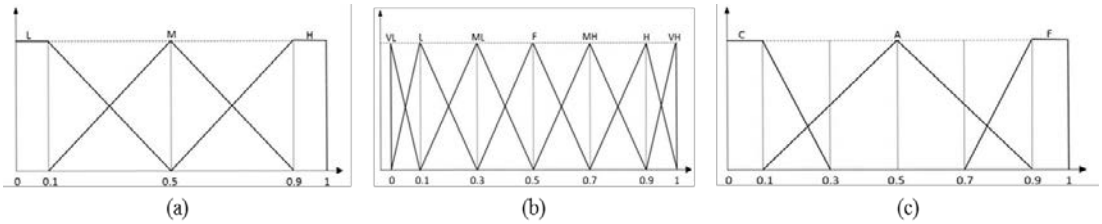


Fig. 6. Fuzzy set of parameters: (a) level of security, (b) memory utilisation and (c) time delay

Table 6. Linguistic scale to evaluate the level of security, memory utilisation and time delay

a) Level of security		
Input Description	**Fuzzy linguistic term**	**Linguistic ratings**
Very Low	VL	[0 0 0.1]
Low	L	[0 0.1 0.3]
Moderate Low	ML	[0.1 0.3 0.5]
Fair	F	[0.3 0.5 0.7]
Moderate High	MH	[0.5 0.7 0.9]
High	H	[0.7 0.9 1]
Very High	VH	[0.9 1 1]
b) Memory utilisation		
Input Description	**Fuzzy linguistic term**	**Linguistic ratings**
Low	L	[0 0 0.1 0.5]
Medium	M	[0.1 0.5 0.9]
High	H	[0.5 0.9 1 1]
c) Time delay		
Input Description	**Fuzzy linguistic term**	**Linguistic ratings**
Close	C	[0 0 0.1 0.3]
Adequate	A	[0.1 0.5 0.9]
Far	F	[0.7 0.9 1 1]

Algorithm 2. Fuzzy TOPSIS for storage server selection

1. Conversion of the linguistic terms to their corresponding triangular/trapezoidal membership scales.
2. Conversion of fuzzy to crisp values (Defuzzification):
   The graded mean representation method is followed for the triangular membership function, whereas the centroid method is used for the trapezoidal membership functions to obtain the appropriate crisp values.
3. Normalization of attribute values:
   The values for different attributes are in different ranges and units after the previous step. Hence, a normalization process, which will convert all the attribute values within the range from 0 to 1 and make them unit less to simplify the decision making, is needed.
   The original sequence for the Higher-the-Better (HB) criterion is normalised as:
   $$\boldsymbol{x_{ij}} = \frac{\boldsymbol{y_{ij}} - \boldsymbol{min}\ \boldsymbol{y_{ij}}}{\boldsymbol{max}\ \boldsymbol{y_{ij}} - \boldsymbol{min}\ \boldsymbol{y_{ij}}} \quad (2)$$
   Meanwhile, the original sequence for the Lower-the-Better (LB) criterion is normalised as:
   $$\boldsymbol{x_{ij}} = \frac{\boldsymbol{max}\ \boldsymbol{y_{ij}} - \boldsymbol{y_{ij}}}{\boldsymbol{max}\ \boldsymbol{y_{ij}} - \boldsymbol{min}\ \boldsymbol{y_{ij}}} \quad (3)$$
   Where, $max\ y_{ij}$ is the highest value of y_{ij} for the *j*th criteria; $min\ y_{ij}$ is the lowest value of y_{ij} for the *j*th criteria; and x_{ij} is the normalised data.
4. Computation of the normalised weighted decision matrix:
   $$\boldsymbol{v_{ij}} = \boldsymbol{x_{ij}} \times \boldsymbol{w_{ij}} \quad (4)$$
   Where v_{ij} the weighted normalised data, and w_{ij} represents the weight of the *j*th criteria.
5. Definition of the fuzzy positive ideal and fuzzy negative ideal solutions:
   $$\boldsymbol{V_j^+} = \{\boldsymbol{v_1^+}, \boldsymbol{v_j^+}, \dots, \boldsymbol{v_m^+}\}$$
   $$\boldsymbol{V_j^-} = \{\boldsymbol{v_1^-}, \boldsymbol{v_j^-}, \dots, \boldsymbol{v_m^-}\}$$
6. Calculation of the distance of each attribute value v_{ij} from a positive ideal value (V_j^+).
   $$\boldsymbol{S_i^+} = \sqrt{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \frac{\boldsymbol{1}}{\boldsymbol{3}} (\boldsymbol{v_{ij}} - \boldsymbol{v_j^+})^{\boldsymbol{2}}} \quad (5)$$
   Where, $i = 1, 2\dots n$; $j = 1, 2\dots m$.
7. Calculation of the distance of each attribute value v_{ij} from the negative ideal value (V_j^-).
   $$\boldsymbol{S_i^-} = \sqrt{\sum_{\boldsymbol{j=1}}^{\boldsymbol{n}} \frac{\boldsymbol{1}}{\boldsymbol{3}} (\boldsymbol{v_{ij}} - \boldsymbol{v_j^-})^{\boldsymbol{2}}} \quad (6)$$
   Where, $i = 1, 2\dots n$; $j = 1, 2\dots m$.
8. Calculation of the relative closeness to the ideal solution:
   $$\boldsymbol{CC_i} = \frac{\boldsymbol{S_i^-}}{\boldsymbol{S_i^+} + \boldsymbol{S_i^-}} \quad (7)$$
   Where, $0 < CC_i < 1$; $i = 1, 2\dots m$; $j = 1, 2\dots n$.
9. Definition of the ranking of the alternatives according to the relative closeness coefficient CC_i in a decreasing order.

4.4.3 Data retrieval

The data retrieval process should also equally retain the security manner as the data storage at different storage servers by maintaining security. This section is further subdivided into

several subsections (i.e. user registration to the data owner, data access request made by user and retrieval of data).

a. User registration

The user has to register himself to the data owner through the CSP to retrieve the data stored in the cloud. In return, a user-ID and a password will be provided to the user. The user will then be registered to the CSP as well. The password will be changed by periodic basis for security concerns.

b. Data request by user

Whenever a user makes a query to access the data in this phase, the query analyzer present at the CSP will check the username provided by the user along with the query. After a legitimate login, the CSP then sends a copy of the request to the data owner. The account will be automatically disabled for a few seconds if the username or the password from the user side is wrong. The page response will also be incrementally delayed after every failed login attempt. After a number of failed attempts, the account will be disabled for an hour or more. After that time, it will be automatically reactivated for further use. These can act as defensive strategies against dictionary or brute force attacks.

i. Request handling at data owner end:

After obtaining the user query, the data owner checks the pre-stored database for that user and generates a session key S_i before directly sending it to the particular user along with the encrypted keyword that the user intends to use. The session key will be composed of various other related information.

$$\text{Session key } S_i = [(DO_{id} \mid Uid \mid A_T \mid Exp\_Time \mid K_{Di}), K_{temp}];$$
$$K_{DNA} = E(K_{ui})$$

Where E = DNA-based encryption;
K_{ui} = keyword that the user is looking for in the query;
DO_{id} = data owner ID registered at the CSP;
Uid = user ID;
Exp_Time = session expiry time limit;
K_{Di} = decryption key for user key (acts up to a limited time);
K_{temp} = secret key shared between the data owner and the CSP for user i.

After generating the session key, the data owner sends it to the corresponding user and notifies the CSP by sharing the temporary secret key K_{temp} using a secure key-sharing scheme. The K_{Di} will contain the information about the primer used during encryption phase to check the integrity of data at receiver end.

ii. Request handling at the CSP's end:

The CSP then asks the user for the key given by the data owner and extracts each composition from the session key through the shared temporary key K_{temp} and checks again for the DOid and Uid with the pre-stored database for assured legitimacy. After that, the CSP checks for the A_T to provide the data access given by the data owner to user 'i' for 'Exp_Time' of seconds. The CSP checks the encrypted index IM′ that points the encrypted keyword K_{DNA} over the completely encrypted document segregated into different security sections (i.e. public, private and limited access) in different storage servers. It also checks the A_T given to the user by the data owner. Finally, all the fragmented parts of the data collected from various storage servers, where they were stored and given access to the user, are embedded based on that CSP. For instance, a user with the grant to access a limited section can likewise permit to obtain the data of the same owner in the private segment.

iii. Data decryption by user:

The user decrypts the data with the decryption key K_{Di} provided by the data owner after obtaining the encrypted data from the CSP.

5. Results and Analysis

5.1 Sensitivity Analysis

A sensitivity analysis is performed to understand the effect of different criterion weights on the selection. The concept applied for the sensitivity analysis is to interchange two criterion weights with each other while keeping the others constant. This analysis shows the behavior of the change in ranking caused by weight manipulation in 16 combinations for six criterions for the present condition. **Table 7** show the overall process of the storage server selection as per the fuzzy TOPSIS algorithm cited in the earlier section with a suitable real-time example. The distance, closeness coefficients and ranking of different storage servers according to the fuzzy-TOPSIS is tabulated in **Table 7 (b)**, where SS4 is found to be in the first rank. All the values of considering parameters are collected in real time. The values of the qualitative parameters, namely level of security, average memory utilisation and average time delay, are taken from **Table 6**. The priority-wise storage servers/storage servers for the selection are as follows: Storage server 4 > Storage server 6 > Storage server 2 > Storage server 5 > Storage server 7 > Storage server 3 > Storage server 1.

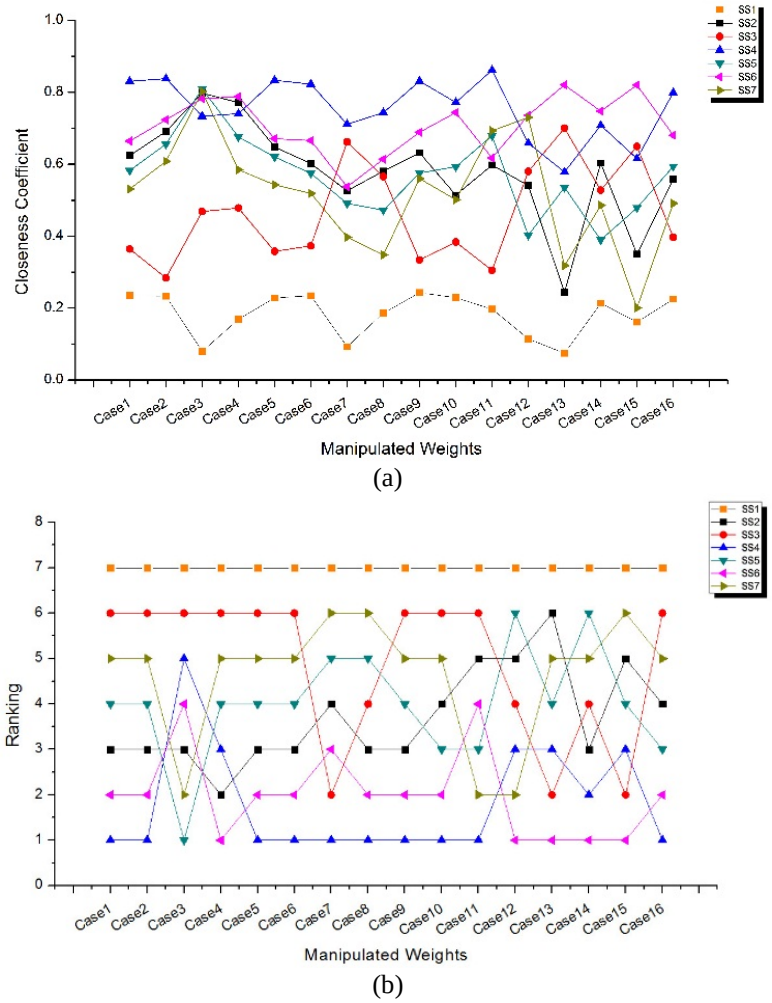
Table 7. TOPSIS table for ranking of storage servers

(a) Initial parameter matrix for the alternative storage servers

Storage Server	Level of security	Avg. Transfer Speed [Mbps]	Avg. Processing speed [GHz]	No. of CPU	Avg. Memory utilisation	Avg. Time delay
SS 1	VL	4	2.5	3	L	C
SS 2	VH	10	3	5	M	C
SS 3	F	16	2	2	H	F
SS 4	MH	12	3.3	7	M	A
SS 5	VH	9	2.60	6	L	A
SS 6	H	10	3.10	5	H	F
SS 7	VH	6	2.10	7	H	C

(b) Calculation of the ranking orders of the storage servers

Storage Server	S_i^+	S_i^-	CC_i	Result - rank
SS 1	0.065	0.020	0.234	7
SS 2	0.031	0.052	0.624	3
SS 3	0.069	0.040	0.364	6
SS 4	0.015	0.074	0.830	1
SS 5	0.037	0.051	0.582	4
SS 6	0.028	0.056	0.664	2
SS 7	0.051	0.057	0.530	5



(a)

(b)

Fig. 7. (a) Sensitivity analysis and (b) ranking of the storage servers after the sensitivity analysis

The CCi and the ranking of all storage servers are calculated for each possible combination using the aforementioned sensitivity analysis methodology. **Fig. 7(a)** represents a change in the closeness coefficient values with weight manipulation. SS4 and SS6 are very close to each other, with high CC_i values most of the time. **Fig. 7(b)** depicts the change in rank, which shows that storage servers have some changes in ranking whenever the weight changes. SS4 and SS6 hold the 1st and 2nd ranks, respectively, whereas SS1 and SS3 possess the last two ranks most of the time under different conditions. SS5 and SS7 drastically change their ranks in different conditions.

5.2 Security Analysis

Fig. 8 shows the security flow of the proposed approach that follows a series of security features to establish a highly secured cloud storage framework. The security strength of the proposed approach is evaluated through three kinds of analysis namely, avalanche effect analysis, correlation coefficient analysis and attack analysis. The proposed encryption strategy is simulated in java platform for its platform independent property and available in-built cryptography functionalities.

5.2.1 Avalanche Effect

Avalanche effect is a highly desirable security measure in good quality cryptosystems. It means that a slight change in the plaintext (hamming distance) or in the key even by a single bit may cause a huge change in the ciphertext. If the change affects half of the bits that means 50% of the ciphertext then it will be treated as a good avalanche effect. The avalanche effect can be observed by following the Equation 8. Here, for a particular plaintext block, the hamming distance is changed randomly up to 5 bit during observation and form five different test cases and found satisfactory results as shown in **Table 8**. **Fig. 9** shows the graphical representation of the avalanche effect for different test cases. After the avalanche effect analysis, the average change in ciphertext is found to be 52.29%.

$$Avalanche\ Effect\ (AE) = \frac{Number\ of\ Changed\ bit\ in\ Ciphertext}{Number\ of\ bits\ in\ ciphertext} \qquad (8)$$

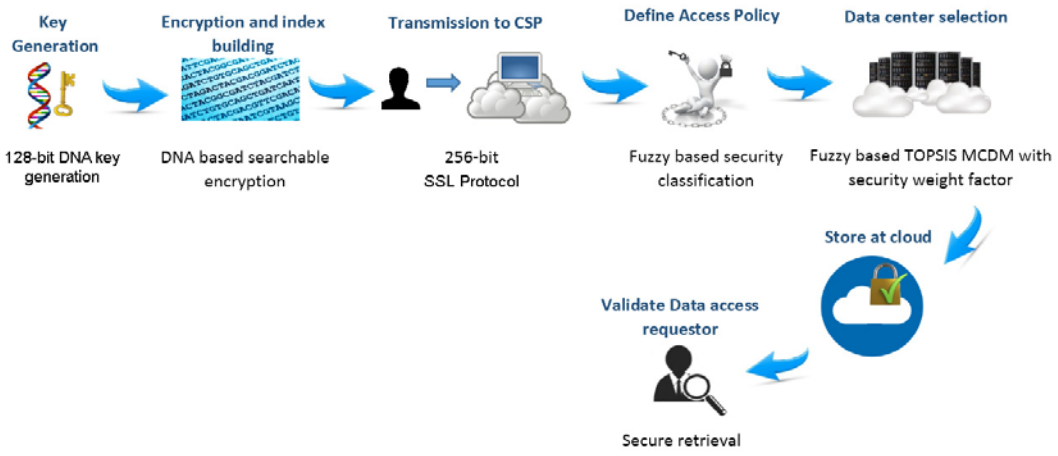


Fig. 8. Security flow of the proposed approach

Table 8. Avalanche Effect Analysis

		Hamming Distance (bit)					
		1	2	3	4	5	
Testcase	1	52.11	55.07	57.29	42.77	55.28	Avalanche Effect (%)
	2	53.72	49.14	50.00	54.10	51.17	
	3	50.00	54.30	52.73	54.30	52.34	
	4	53.51	50.09	55.86	53.91	54.29	
	5	50.72	53.91	50.00	48.72	51.93	

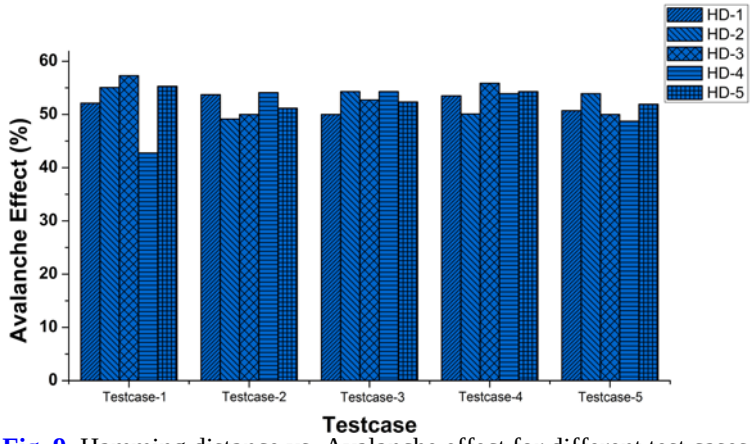


Fig. 9. Hamming distance vs. Avalanche effect for different test cases.

5.2.2 Correlation Coefficient

Correlation coefficient is considered as one of the important aspects of the security of block ciphers that deal with a dependency of the individual output bits on the input bits. This paper used Correlation coefficient to measure the dependency between plaintext and ciphertext. The correlation values can determine the confusion effect of the block cipher. It is the measurement of the degree of linear relationship between two variables. Correlation coefficient is a number between (-1) and (1), where 1 indicates a strong positive relationship, -1 indicates a strong negative relationship and a result with zero indicates no relationship at all. Equation 9 represents the formula through which the correlation coefficient effect can be observed where, x specifies first score, y specifies second score, Σxy specifies sum of the product of first and second scores, Σx specifies sum of first scores, Σy specifies sum of second scores and Σx2 specifies sum of square first scores.

$$\boldsymbol{Correlation}\ (r) = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}} \qquad (9)$$

A comparative analysis of correlation coefficient values of the proposed encryption approach with the traditional AES for five testsets are shown in **Table 9**. The values near to zero represents better result and it has been observed that for all the testsets the proposed approach outperforms the AES. **Fig. 10** shows the correlation coefficient test results between the proposed encryption approach and the AES.

Table 9. Correlation Coefficient analysis

Testcase	Proposed Algorithm	AES
1	-0.0551	-0.1662
2	-0.1021	-0.1722
3	-0.1003	-0.2382
4	0.0772	0.2239
5	0.1561	-0.3098

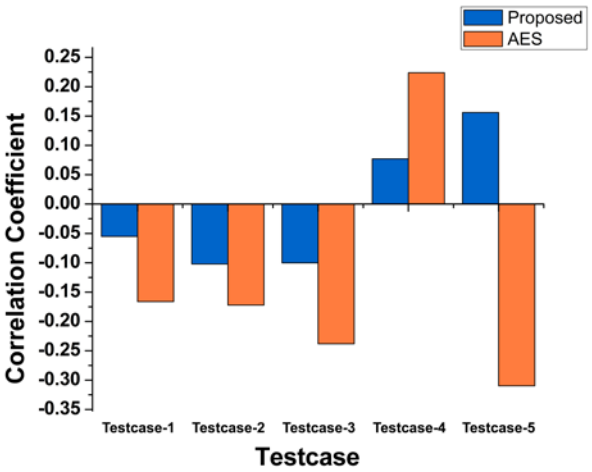


Fig. 10. Correlation Coefficient analysis for AES with proposed approach.

5.2.3 Attack Analysis

The attack analysis of the proposed approach has been illustrated below, where the process of defense against various attacks has been explained.

5.2.3.1 Brute-force Attack

The 128-bit randomly chosen key is used for the key generation. Thus, there would be 2^{128} (i.e. 3.403×10^{38}) numbers of possible combinations of keys for cryptanalysis if an attacker tries to apply the Brute-force attack on the ciphertext. Moreover, a combination of four DNA bases is randomly chosen from 256 combinations, leading to 256 numbers of combination strings. A total of 2^{256} (i.e. 1.157921×10^{77}) numbers of possible key combinations must be tried for cracking the 256-bit SSL session key. Thus, there are a total of $2^{128} \times 256 \times 2^{256} = 1.009 \times 10^{118}$ numbers of possible keys for the Brute-force attack. In conclusion, from the above large key size, cryptanalysis is almost impossible. Moreover, approximately 163 million DNA sequences are publicly available. Thus, the probability of an attacker to crack the primer padding in the DNA encryption process and make a successful guess on the reference DNA sequence used in the DNA-encoding step is $1 / (1.6 \times 10^{8})$. In other words, guessing the correct primer from this sea of possible DNA sequences is almost impossible.

5.2.3.2 Dictionary Attack

The dictionary attack tries to match the password with most occurring words or words of daily life usage. Various automated tools can guess it very easily if users choose a very common password by themselves. As a result, the user's account will be compromised. However, in the proposed approach, the CSP is responsible for providing both the username and the password to the user. Moreover, an incremental delay strategy is followed for every failed login to restrict the automated tools to attack.

5.2.3.3 Side Channel Attack

In this type of attack, an attacker takes advantage of a shared physical component to steal information from the victim. Any co-resident user registered in the same CSP can perpetrate a side channel attack. In the proposed approach, the owner's data are not stored in the CSP end,

but in different geographically distributed storage servers used to store the actual data in a fragmented manner. Moreover, the data stored in the storage servers are in a strongly encrypted form, and neither the CSP nor the storage servers can retrieve them.

5.2.3.4 Known Plaintext Attack

In this type of attack, attacker has the knowledge about some random set of plaintexts and the corresponding ciphertexts. The main objective behind this attack is to approximate the operation used in the encryption process. During this, an attacker tries to make a linear relationship between the plaintext and their corresponding ciphertext. For example, using the Equation 10 to figure out the probability of the equation to be satisfied.

$$\boldsymbol{x_1 \oplus x_2 \oplus \ldots \oplus x_n \oplus y_1 \oplus y_2 \oplus \ldots \oplus y_m = 0} \qquad (10)$$

In the above equation, x_i represents the bit values in a random plaintext and y_j represents their corresponding ciphertext bit values. For an encryption process, if the probability of satisfying the Eq. (10) is much greater than or smaller than ½ that means 0.5, then it will be treated as vulnerable and can be possible for linear cryptanalysis with less number of known plaintexts. The probability deviation from the value ½ is called probability bias. A bias value near to ½ represents better security against linear attack. **Table 10** lists a sample linear probability analysis of the proposed encryption approach performed for 5 different random linear expressions formed with randomly selected bit values of input key sequence K_{DNA} and the resultant final cipher key K_{fin}.

For an example, the linear probability observed in proposed approach for Linear Expression-3 is $(1.6639 \times 10^{38})/(3.4028 \times 10^{38}) = 0.489$ approx. and the probability bias value is $|0.489 - 0.5| = 0.011$ which is close enough to the desired value that means ½. Therefore, the proposed key generation technique provides better security against the known plaintext attack or linear attack.

Table 10. Linear probability analysis

Linear Expression	Linear Probability	Probability Bias
1	0.438	0.062
2	0.417	0.083
3	0.489	0.011
4	0.50	0.00
5	0.518	0.018

5.2.3.5 Threat from the CSP

We analyze the security of the communication channel between the user and the CSP and between the data owner and the storage server. The undaunted faith on the CSP is not right. A situation, where a CSP can betray the data owner by spilling data to the adversary parties, may arise. For this, the ideal arrangement used in the proposed model is the DNA-based data encryption at the data owner's end before uploading to the cloud. In addition, using the SSL handshaking over public internet during data transmission between the data owner and the CSP makes the approach much secure. The followed 256-bit encryption also guarantees the data security in an improved manner.

5.2.3.6 Man- in- the- Middle Attack (MITM)

A man-in-the-middle attack can succeed only when it is possible for an attacker to mimic every endpoint of the transmission. In the proposed approach, both authentication and encryption are followed to secure the network against MITM attacks. The 256-bit SSL is used for authenticating one or both parties using a mutually trusted certification authority. Using a DNA-based encryption algorithm is also another effective method of defending against MITM attacks.

5.2.3.7 Threat from Storage Server

The CSP and the storage server in the proposed approach are separate and geographically distributed. The data are fragmented into several parts, and then stored into a number of storage servers or storage servers, which are independent and unknown to each other. Moreover, the data stored in the storage servers are already encrypted twice. Hence, a storage server can clearly never crack the encrypted data because the data stored in a particular storage server are just a part of the whole. It can also never know the locations of the rest of the parts. In addition, the proposed model provides a qualitative analysis of different security parameters to segment the whole data into different levels of security as per user necessities. Hence, this approach assures a secured data storage and ensures the security of data during transmission.

5.2.4 Functionality Analysis

An efficient cloud data storage security model should have the capacity to defeat all the conceivable storage issues of cloud computing to achieve its extreme statures and prevent the owner's data from all kinds of attacks and threats.

Table 11. Feature comparison of various cloud based secured storage framework

	Features	**Wang et al. [7]**	**Prasad et al. [14]**	**Cao et.al [4]**	**Xia et al. [19]**	**Cui et al. [20]**	**Proposed**
1	Identity and authentication	✓	✓	✓	✓	✓	✓
2	Authorization	✓	✓	✓	✓	✓	✓
3	Confidentiality	✓	✓	✓	✓	✓	✓
4	Indexing of data	×	×	✓	✓	×	✓
5	Keyword search	×	×	✓	✓	×	✓
6	Dynamic storage server selection	×	×	×	×	×	✓
7	Data fragmentation	×	×	×	×	×	✓
8	Multiple independent storage servers	×	×	×	×	×	✓

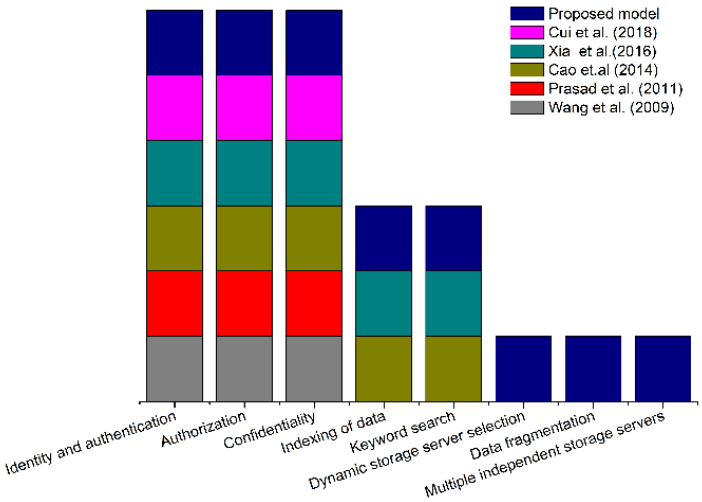


Fig. 11. Feature analysis graph

Table 12. Comparative analysis of various encryption approach

	State of the art				Literatures				
Factors	**DES**	**3DES**	**AES-128**	**Blowfish**	**Sadeg et al. [26]**	**Majumder et al. [33]**	**Kar et al. [25]**	**Majumdar et al. [34]**	**Proposed**
Key Length (bits)	56	168	128	448	128/256	256	16	256	128
Block Size (bits)	64	64	128	64	128	256	64	256	128
Avg. Entropy (per byte of encryption)	2.95	2.94	3.84	3.93	3.33	3.44	2.82	3.51	4.603
Avg. Avalanche effect (%)	41.10	40.44	44.53	40.09	41.14	46.73	39.02	43.24	52.29

Table 11 shows a comparison of the proposed model with the other cloud storage security-related models. It can be noticed that most of the approaches were concerned about authorization, authentication and confidentiality. Only Cao and Xia's approaches had a searchable encryption and encrypted index security feature like our proposed model. However, in our approach, the proposed DNA-based encrypted index and data provide a more secured data storage and a faster data access. In addition, our approach provides a powerful environment, where data security is assured at an enhanced level, by introducing features such as data level of security, dynamics server selection, data fragmentation and multiple independent storage servers etc. **Fig. 11** shows the comparison of various cloud based secured storage framework with the proposed model with respect to various security features. Moreover, a comparison between different standard encryption approaches with respect to

various security factors has been listed in **Table 12**. From these comparisons, it can be realized that the proposed model covers many of the possible security concerns by providing different security approaches that are adequate to manage the security issues of the cloud computing storage.

6. Conclusion

This work presents a framework that ensures a solution to the many data security issues in cloud storage. In addition, features such as data dissection, searchable encrypted index and strong DNA-based encryption along with a robust key generation process make it perfect to store data in the cloud. The proposed approach provides effective data storage criteria in dynamically selected secured and functionally appropriate storage servers from the diverse geographical network and has a leading advantage over the other methods proposed till date. After different security, sensitivity and functionality analyses, the strength of the proposed scheme has been concluded.

References

[1] Gartner, "Newsroom: Consumers Will Store More Than a Third of Their Digital Content in the Cloud by 2016," *Press Release*, 2012.
[2] A. T. Velte, T. J. Velte, R. C. Elsenpeter and R.C. Elsenpeter, "Cloud Computing: A Practical Approach," *New York: McGraw-Hill*, pp. 44, 2010.
[3] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. of International Conference on Financial Cryptography and Data Security, Springer Berlin Heidelberg*, pp. 136–149, January, 2010. Article (CrossRef Link).
[4] N. Cao, C. Wang, M. Li, K. Ren and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 1, pp. 222–233, 2014. Article (CrossRef Link).
[5] R. Chen, Y. Mu, G. Yang and F. Guo, Wang, X, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 789–798, 2016. Article (CrossRef Link).
[6] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *International Conference on the Theory and Application of Cryptology and Information Security, Springer Berlin Heidelberg*, pp. 577–594, December 2010. Article (CrossRef Link).
[7] C. Wang, Q. Wang, K. Ren and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. of IWQoS*, pp. 1–9, July, 2009. Article (CrossRef Link).
[8] Q. Wang, C. Wang, K. Ren, W. Lou and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011. Article (CrossRef Link).
[9] R. Buyya and M. Murshed, "Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing," Concurrency and Computation: Practice and Experience, vol. 14, no. 13–15, pp. 1175–1220, 2002. Article (CrossRef Link).
[10] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer*, vol. 38, no. 5, pp. 32–38, 2005. Article (CrossRef Link).
[11] C. Balding, "Is Your Amazon Machine Image Vulnerable to SSH Spoofing Attacks?," *Cloud Security*, 2008.
[12] H. Takabi, J. B. Joshi and G. J. Ahn, "Securecloud: Towards a comprehensive security framework for cloud computing environments," in *Proc. of 34th Annual Computer Software and Applications Conference Workshops (COMPSACW), IEEE*, pp. 393–398, July 2010.
Article (CrossRef Link).

[13] S. Venkatesan and A. Vaish, "Multi-agent based dynamic data integrity protection in cloud computing," in *Proc. of International Conference on Advances in Communication, Network, and Computing, Springer Berlin Heidelberg*, pp. 76–82, March 2011. Article (CrossRef Link).
[14] P. Prasad, B. Ojha, R. R. Shahi, R. Lal, A. Vaish and U. Goel, "3 dimensional security in cloud computing," in *Proc. of Computer Research and Development (ICCRD), 2011 3rd International Conference on IEEE*, pp. 198–201, March 2011. Article (CrossRef Link).
[15] S. K. Sood, A.K. Sarje and K. Singh, "A secure dynamic identity based authentication protocol for multi-server architecture," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 609–618, 2011. Article (CrossRef Link).
[16] S. K. Sood, "A combined approach to ensure data security in cloud computing," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1831–1838, 2012. Article (CrossRef Link).
[17] J. Spillner, J. MüLler and A. Schill, "Creating optimal cloud storage systems," *Future Generation Computer Systems*, vol. 29, no.4, pp. 1062–1072, 2013. Article (CrossRef Link).
[18] B. Dong, Q. Zheng, F. Tian, K. M. Chao, R. Ma and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 184–1862, 2012. Article (CrossRef Link).
[19] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE transactions on parallel and distributed systems*, vol. 27, no. 2, pp.340-352, 2016. Article (CrossRef Link).
[20] H. Cui, R. H. Deng, and Y. Li, "Attribute-based cloud storage with secure provenance over encrypted data," *Future Generation Computer Systems*, vol. 79, pp.461-472, 2018. Article (CrossRef Link).
[21] M. R. Abbasy and B. Shanmugam, "Enabling data hiding for resource sharing in cloud computing environments based on dna sequences," in *Proc. of the 2011 IEEE World Congress on Services, SERVICES'11*, pp. 385–390, 2011. Article (CrossRef Link).
[22] A. Khalifa and A. Atito, "High-capacity DNA-based steganography," in *Proc. of 8th International Conference on Informatics and Systems (INFOS2012), IEEE*, pp. BIO-76-80, 2012. Article (CrossRef Link).
[23] A. Atito, A. Khalifa and S. Z. Rida, "DNA-based data encryption and hiding using playfair and insertion techniques," *Journal of Communications and Computer Engineering*, vol. 2, no. 3, pp. 44, 2012.
[24] S. Pramanik and S. K. Setua, "DNA cryptography," in *Proc. of 7th International Conference on Electrical & Computer Engineering (ICECE), IEEE*, pp. 551–554, December 2012. Article (CrossRef Link).
[25] N. Kar, A. Majumder, A. Saha, S. Deb and M. C. Pal, "Data security and cryptography based on DNA sequencing," *International Journal of Information Technology & Computer Science (IJITCS)*, vol. 10, no. 3, 2013.
[26] S. Sadeg, M. Gougache, N. Mansouri and H. Drias, "An encryption algorithm inspired from DNA," in *Proc. of International Conference on Machine and Web Intelligence (ICMWI), IEEE*, pp. 344-349, October 2010.
[27] X. Wang and Q. Zhang, "DNA computing-based cryptography," in *Proc. of Fourth International Conference on Bio-Inspired Computing, BIC-TA'09. IEEE*, pp. 1–3, October 2009. Article (CrossRef Link).
[28] A. Gehani, T. H. LaBean and J. H. Reif, "DNA-based Cryptography," in *Proc. of 5th DIMACS Workshop on DNA-based Computers*, pp. 167-188, 1999. Article (CrossRef Link).
[29] S. Marwan, A. Shawish and K. Nagaty, "DNA-based cryptographic methods for data hiding in DNA media," *Biosystems*, vol. 150, pp. 110–118, 2016. Article (CrossRef Link).
[30] L. Pray, "Discovery of DNA structure and function: Watson and Crick," *Nature Education*, vol. 1, no. 1, pp. 100, 2008.

[31] M. Toorani and A. Falahati, "A secure variant of the Hill cipher," in *Proc. of 2009 IEEE Symposium on Computers and Communications*, pp. 313-316, 2009. Article (CrossRef Link).
[32] T. L. Saaty, "The analytic hierarchy and analytic network processes for the measurement of intangible criteria and for decision-making," *Multiple criteria decision analysis: state of the art surveys, Springer, New York, NY*, vol. 78, pp. 345-405, 2005. Article (CrossRef Link).
[33] A. Majumder, A. Majumdar, T. Podder, N. Kar and M. Sharma, "Secure data communication and cryptography based on DNA based message encoding," in *Proc. of International Conference on Advanced Communication Control and Computing Technologies (ICACCCT), IEEE*, pp. 360–363, May 2014. Article (CrossRef Link).
[34] A. Majumdar, T. Podder, A. Majumder, N. Kar and M. Sharma, "DNA-based cryptographic approach toward information security," in *Proc. of Intelligent Computing, Communication and Devices, Springer India*, pp. 209–219, 2014. Article (CrossRef Link).

Abhishek Majumdar received his B. Tech. degree in Computer Science and Engineering from Tripura Institute of Technology, Narsingarh, India in 2012. He received his M. Tech. degree in Computer Science and Engineering from Punjab Technical University, India in 2014 Currently, he is pursuing his Ph.D. from National Institute of Technology Silchar, India under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation. His research areas are cloud computing, optimization, information security and machine learning.

Arpita Biswas received her B. Tech. degree in Computer Science and Engineering from Tripura Institute of Technology, Narsingarh, India in 2012. Punjab Technical University, India in 2014. Currently, she is pursuing her Ph.D. from National Institute of Technology Silchar, India under the Visvesvaraya PhD Scheme of Ministry of Electronics & Information Technology, Government of India, being implemented by Digital India Corporation. Her research areas are information security, integration of IoT with cloud computing and optimization.

Dr. Krishna Lal Baishnab received his M.Tech from Indian Institute of Technology kharagpur, WB and Ph.D. from National Institute of Technology Silchar, India. He is currently working as associate professor, Electronics and Communication Engineering, NIT Silchar. He has more than 20 years of teaching experience and published more than 50 articles in various reputed journals and international conferences. His research areas are Analog VLSI and RF Design: Design of continuous/ discrete time Analog circuit for Bio-medical applications, RF CMOS circuit design and wireless systems.

Dr. Sandeep K. Sood received his Ph.D. in Computer Science & Engineering from IIT Roorkee, India. He is currently working as Associate Dean (A.A. & S.W), Head & Professor, Computer Science & Engineering, G.N.D.U. Regional Campus, Gurdaspur. He has 17 years of teaching and 9 years of research experience. He has more than 50 research publication. His work is published and cited in highly reputed journals of Elsevier, Springer, Wiley and IEEE. His citation number according to Google Scholar is 1350 with h-index equal to 18 and i10-index equal to 37. His research areas are cloud computing, IoT, big data analytics, information security, cyber physical system.