# Design A Scds-Tm Model For Managing Data In Cloud Based System

**Hira Lal[1] and Dr. Arpana Bharani[2]**

[1,2]Department of Computer Science, Dr. A. P. J. Abdul Kalam University, Indore (M.P.) - 452010

**Abstract:**
In the present research work a SCDS-TM model has been proposed as a solution to the problem of data secrecy in cloud architecture. SCDS-data TM's privacy issues and potential remedies will be examined in the paper. Using this model, it is possible to store and retrieve data in a secure manner. The proposed framework is compared to existing models in terms of a number of metrics. It's possible to accomplish data secrecy, security, and safe retrieval all at once utilising this economical paradigm.

**Keywords :** SCDS-TM model, Cloud Based System, Privacy Issue and Data Security

## 1. INTRODUCTION

New technologies and the arrival of the internet, e-commerce applications, and social networks have resulted in an enormous amount of data being generated every day. Cloud computing has grown as a platform for outsourcing and providing data services. Businesses may store as much data as they need for as little money as possible with this service. Sensitive data hosted in the cloud raises a slew of privacy and security concerns. Sensitive data, such as government or business documents and medical records, cannot be stored on the cloud unencrypted. Many people's interest may be piqued, leading to worries about their privacy. In light of this fact, a new structure and set of regulations are needed to safeguard cloud data. In order to safeguard cloud data, data encryption may be used, however this comes at a price. However, it is likely to be difficult to retrieve information in these scenarios. Cloud-based data storage and retrieval will become the standard in the near future. A broad variety of ways and processes were used to get information. CSPs may find it difficult to retrieve data from the cloud if all of these solutions focus on protecting data held in a third-party province [1].

Malicious data tampering attacks and even server collusion attacks may be conceivable despite the homomorphic scheme's superior efficiency and resilience to scheming failure. However, malicious users may find a way to avoid informing the cloud storage's owner when unauthorised access happens, making cloud storage end-to-end security a difficulty. Since the previous model did not handle data confidentiality, integrity, or retrieval, a SCDS-TM model has been developed for the cloud.

Using the SCDS-TM paradigm, secret data is not only stored and retrieved securely; storage servers and key servers are also effectively coordinated. Dynamic data operations such as insertion or deletion of blocks may be performed using this model, which validates the integrity of the data blocks by using MHT construction [2].

To make sure that data is complete and up-to-date, MHT employs a multi-step process. Added to that, the SCDS-TM framework's simplicity, efficiency, and strategy for public verifiability provide public verifiability without jeopardising the privacy of data owners. Data confidentiality and integrity must be seamlessly integrated into the protocol architecture in order to provide an elegant cloud storage system that enables efficient data storage, data integrity, and data retrieval.
This SCDS-TM supports the distributed storage system, making it possible to store and retrieve data in a secure manner. A SCDS-TM model is offered here for quick data retrieval. Using an elliptic curve encryption method and coordinate matching, sensitive data may be recovered.

## 2. ARCHITECTURE AND WORKFLOW OF SCDS-TM MODEL

Data privacy and security were major concerns while it was being stored, as was ensuring the data's integrity. Fig. 1 displays the architectural design of the proposed SCDS-TM model. The standard for cloud servers is a "honest–but-curious" attitude. Even if the server is telling the truth, it is curious about the information it has. As a consequence, the cloud storage architecture is comprised of storage servers and key servers. Because of the system's great degree of dispersion, storage and key servers perform their own tasks [3]. There are two types of servers: those that store data and those that deal with sensitive information..

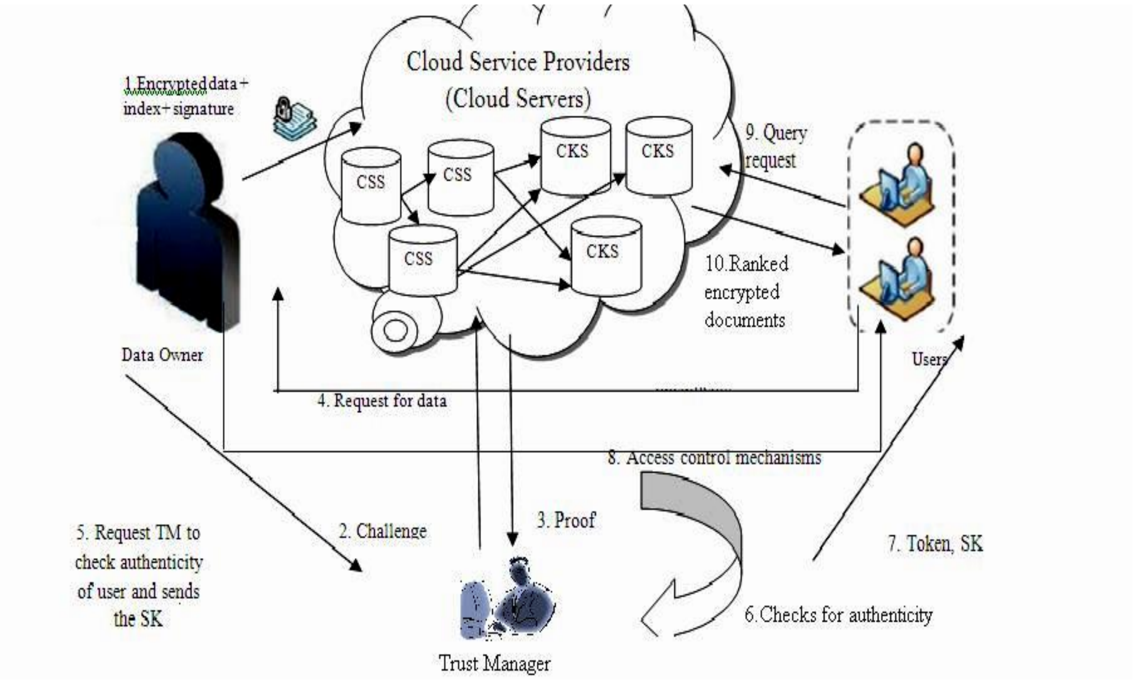

**Figure 1: Architecture of SCDS-TM model**

Each component of the SCDS-TM model is independent of the others: Users, as well as the Trust Manager (TM), are in accord. All of the user's vital data may be saved on the cloud storage servers by the data owner themselves. The trust manager has the same degree of expertise as the owner. The TM's principal objective is to verify the user's identification. A TM will make it easier for owners to react to and authenticate user requests. To ensure appropriate data storage, a trust manager must challenge the cloud server after the data owner has transferred his data to the server. TM is in charge of all cloud server operations, including user requests [4].

Elliptic curve cryptography is used to encrypt data using ECC Encryption technique, which is represented in Fig. 1. The IndexGen algorithm and the SigGen algorithm are used to create an index and signature for these blocks. The owner of the CSS additionally maintains a Cloud Key Server where the accompanying index and signatures are stored, in addition to encrypting and storing the data there (CKS). In order to ensure that encrypted data, an index, and a signature are stored in the cloud correctly, a trust manager uses the chal message. [5] The cloud server constructs the proof (Pr) after confirming the correctness of the owner's data in the cloud storage using the ProofGen method. On a case-by-case basis, either True or False is returned to the original owner. Using the Execute Update algorithm, the cloud server will perform an update if the data owner asks it. An algorithm known as Prove Update is used by the trust management to ensure that data is always up-to-date [6].

It is required that the trust manager authenticate the user's identity when a user requests access to their data. Users are given tokens and secret keys (SK) and are told whether they have been authenticated. As soon as the trust manager receives the authentication information, the database owner sends the procedures for controlling who may access the data. A user with access control rights may read, write to, and execute the owner's data [7] and data retrieval.

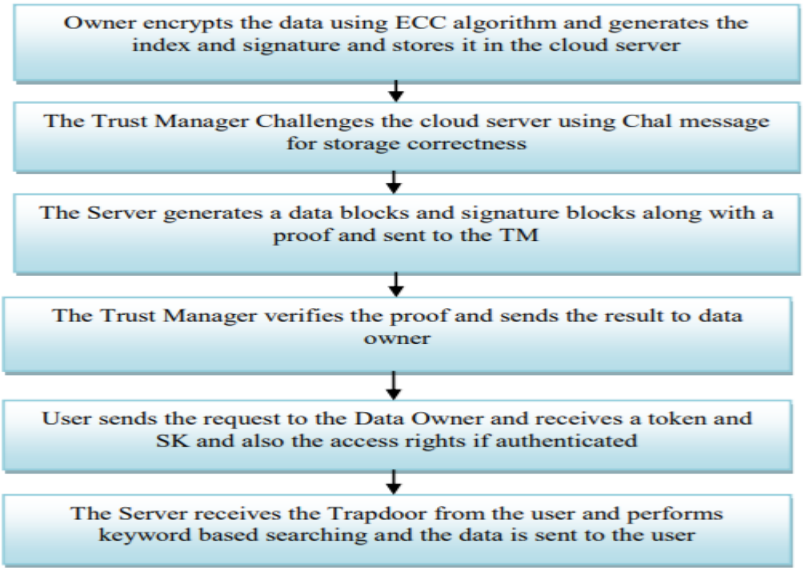
Owner encrypts the data using ECC algorithm and generates the index and signature and stores it in the cloud server
The Trust Manager Challenges the cloud server using Chal message for storage correctness
The Server generates a data blocks and signature blocks along with a proof and sent to the TM
The Trust Manager verifies the proof and sends the result to data owner
User sends the request to the Data Owner and receives a token and SK and also the access rights if authenticated
The Server receives the Trapdoor from the user and performs keyword based searching and the data is sent to the user

**Figure 2: Workflow Diagram of SCDS-TM Model**

After obtaining the necessary permissions from the data owner, the user sends the request to the cloud server for execution. As soon as it gets the query, the cloud server starts searching. Using keywords in a search is a way to speed things up. With the TrapGen approach, a dictionary of keywords is stored on the cloud server and may be exploited by hackers. After the trapdoor has been produced, the user makes a query request. This includes the URL for trapdoor (trapdoor, keyword, index). Index, keywords, and the trapdoor are all used by the cloud server to conduct searches that return just the top-k most relevant results. An elliptic curve decryption technique is used by the user to decipher encrypted material [8]. Under the SCDS-TM paradigm, owners' communication and computation costs have been considerably reduced. Additionally, it is the user's responsibility to do a keyword-based search in order to locate and get the necessary documents. Fig. 2 depicts the proposed SCDS-TM model process flow diagram.

Data transmission rate, data security, query execution time, probability ratio, and accuracy ratio are some of the performance metrics utilised to evaluate the proposed SCDS-TM model's performance. The Trust Manager's involvement reduces the owner's Communication Overhead significantly. It is important to know how fast data is transported from the cloud server to the user as well as how the data is successfully transmitted to the user without any loss from the cloud server while using a cloud server.

The length of time it takes a cloud server to perform a particular query or request is measured by Query Execution Time. Cloud server's Search Precision and Probability Ratio may be used to evaluate how probable it is that a user will find the message they are looking for after searching the cloud server's database [9].

## 3. MATHEMATICAL MODELING

The mathematical modeling of the proposed method is explained below:

### 3.1 Key Gen ($1^k$)

Let the owner of the file F partition the file into distinct data blocks like F1, F2, F3.........Fn; where ni Zp and p is always a prime. ni is the number of data blocks in the file. Invoking the Key Gen algorithm generates both the owner's private key and the public key. It is the owner's responsibility to produce the key pairs (opk, opuk). Create the value by selecting Z p and entering the number. Sk=(, opk), and puk=(puk) are the secret and the public keys respectively ( ,opuk). With these two keys as a foundation, the owner may create the master key mk=(, ).

**3.2 Encryption ():** Elliptic Curve Cryptography is used to encrypt the message M. It is necessary to produce the keys for the owner before encrypting the data. KeyGen() is used to create the owner's keys.

**C←ECC Encryption (M, r, P, BP)**

Input: The message M, the base point BP, r a random number, owners‟ secret key sk and a elliptic curve point P.

Output: The encrypted message i.e the cipher text C

- The message M is mapped with the elliptic curve at a point P.
- A random integer r is chosen such that r€ [1,n-1]
- The encrypted message (cipher text) is determined as a pair:

C= {(r*BP)+($P_m$+(r*sk))a

### 3.3 T←Tag Gen(C,sk)

When creating tags, the algorithm takes the data file F and the owner's secret key into account. An element is selected and a tag is generated based on the encrypted data file F (C=(C1,...,Cn)) where C=(C1,...,Cn). The tag contains the following:

name||n|| $y$||SSig$_{sk}$(name||n||$y$)

### 3.4 τ←Index Gen (F, mk)

An index for all of the encrypted documents is constructed using K-Nearest Neighbour computing technique based on an index built using a binary data vector Vi for each document included in data file Fi. Vi[j] represents the keyword associated with each document in data file Fi. One (e+1)-bit vector (X) and two (e+1)*(e+1) invertible matrices (A1, A2) typically comprise the master key mk, where e denotes the number of fields in each record (ri). Every encrypted document Ci is represented in the index by the two matrices A1 and A2; where A1 and A2 are matrices for each encrypted document. C=($c_1$,$c_2$,….,$c_n$).

### 3.5 γ ←Sig Gen (sk,F)

After generating the tag for the file F, the owner computes the signature γ for each data blocks $n_i$(i=1,2,…. ,n) as;

$$\gamma i \leftarrow \{H(n_i)_{.}\ y^{ni}\}^{\alpha}$$

the data file F. and his secret key sk There are a total of n signatures for each and every data block. The owner, with the aid of MHT, creates a root R with the leaf node occupying the ordered collection of file tags after producing the signature for each data block. The papers are then signed using the owner's private key.

α: sig$_{sk}$(DH(R) )← (DH(R) )$^{\alpha}$ .

Signed files and signature sets are sent to a cloud server by the owner and deleted from the owner's local storage once the private key has been used to sign it.

### 3.6 (Pr)←Proof Gen(F,chal, ?)

In order to produce a proof of storage for the data, the cloud server uses this method. A data integrity proof (Pr) is generated for the blocks indicated in the challenge chal by receiving the data file F, its signature, and the challenge chal as input parameters. The Trust Manager generates chal by selecting a random element subset I=s1....sc; where each I is I and ui is B Zp. To begin with, the cloud server produces the data blocks and signature blocks after receiving the challenge.

$\mu = \sum u_i \, n_i \in Z_p$

$\sigma = \prod \sigma_i^{ui} \in B$

Where $i = s_1 \ldots\ldots s_c$ and μ represents the data blocks and σ represents the signature blocks.

### 3.7 (True, False)←Proof Verf (puk,c hal, Pr)

It is utilised by the cloud server's owner to verify the created evidence. It accepts the public key puk, the challenge chal, and the proof Pr as input arguments and returns yes or false depending on the input.

### 3.8 (P’,F’,$\mathbf{Pr_{update}}$) ←ExecuteUpdate(F, P,update)

Data file F and signature set are accepted as input parameters and an operation called "update" is requested as an output parameter. The cloud server uses these parameters to call this procedure and returns the updated data file F, signature set, and proof in response.

### 3.9 {(True, False, $\mathbf{sig_{pk}}$(DH(R’))} ←Prove Update (puk, update, $\mathbf{Pr_{update}}$)

When the cloud server generates an update, it is authenticated (verified) using this technique (verifier). True or False is outputted depending on the verification.

### 3.10 $\boldsymbol{D^{\frown}}$←TrapGen(D)

With k keywords of interest , and D be the dictionary of the keywords set that consists $D=(D_1,D_2,\ldots\ldots D_n)$ ,this algorithm generates the trapdoor $D^{\frown}$;

### 3.11 $\boldsymbol{F^{\frown}}$←Query ( $\boldsymbol{D^{\frown}\; k\; \tau}$)

Searching is carried out on an index and via a trapdoor to build a list of all of the top-k documents, ordered according to their similarity value, by cloud server when it gets query requests as (from the users).

### 3.12 M ←Decryption(C, upk, P)

Input: The coordinate point C, users‟ private key upk , owners‟ secret key sk and elliptic curve point P.

Output: The original message M

Extract the x co-ordinate of the encrypted message (cipher text) C as,

- x=xcod(C) ; where xcod is the function to compute the x co-ordinate.
- Similarly extract the y co-ordinate of the encrypted message C as,
- y=ycod(C) ; where ycod is the function to compute the y co-ordinate.

Compute Φ, where Φ=x*y

- Identify the mapped point P by calculating $\{(P_m+( r*sk))\ \Phi)\}/upk$
- The original message M is extracted by un-mapping the point P.

## 4. STEPS AND PROTOCOL DESIGN IN SCDS-TM MODEL

### 4.1 Steps in SCDS-TM Model

i. F1 through Fn, where ni=Zp, is a set of data blocks that make up the data File F, and each block is assigned a unique number. ECC Encryption () technique is used to encrypt the owner's personal data using elliptic curve cryptography. By using IndexGen () and SigGen () algorithms, the owner may then build an index and a signature for the encrypted data [10].

ii. The cloud storage servers hold the encrypted data (C, index, and signatures). The cloud key servers are where the encryption keys are kept. A chal message is used by the Trust management to verify that the server is storing data correctly.

iii. The cloud server in turn creates the data blocks and signature blocks and also a proof for the challenged blocks and transmits this evidence to the TM. It is now up to TM to run the ProveUpdate () algorithm and return either True or False as a consequence of this process, which was completed using ProofVerf() and ExecuteUpdate(), respectively, on the cloud server's evidence.

iv. The user asks the owner for the information.

v. Upon receiving the request, it is in-turn passed to the Trust Manager(TM) for confirming the authenticity of the user and the validity of the block tags are validated by executing the relevant MHT algorithm.

vi. The TM verifies the identity of the person using it.

vii. If authorised, transmits a token to the user with the secret key required to decode the message; otherwise the request is refused.

viii. The user obtains access permissions from the owner and uses them to submit a search request or a Trapdoor to the cloud server through an appropriate access control mechanism.

ix. When a server receives an encrypted trapdoor, it does a ranked search of relevant documents based on the index and delivers the top k-relevant encrypted files.

x. By means of the secret key acquired from the TM, the user decrypts the encrypted documents and the correct data is retrieved.

### 4.2 Protocol Design of SCDS-TM Model

In this part, we'll go through the proposed framework's data flow diagram in more detail. Fig. 3 shows the storage process in three phases: Owner Initialization, Data Confirmation, and Verification or Auditing.

**Phase 1:** In the initial setup of the system the owner produces the encryption keys and computes the index and signature for the various data blocks that are to be encrypted.

**Phase 2:** In order to ensure that data stored in the cloud is accurate, the data owner seeks the help of the trust manager, who performs an audit of the cloud server. Data is deleted from local storage when a user confirms their identity. As follows, the auditing process involves back-and-forth communication between both parties:

- The cloud server receives a chal(i, ui) s1i sc; message from the trust manager, which uses the chal() algorithm to construct a chal message for the data blocks in file F.

- Data blocks and signature blocks are computed, and a proof is generated and sent to the trust management after receipt of the "challenge" message from that manager.

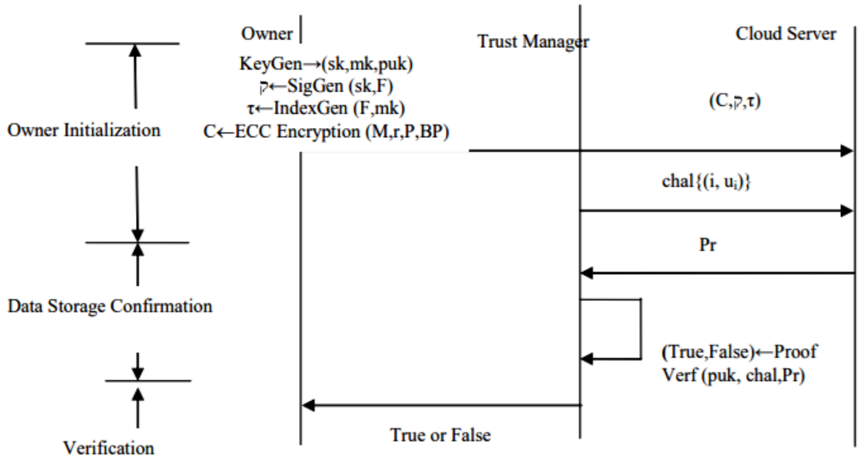

**Figure 3: Auditing Protocol of SCDS-TM Model**

**Phase 3:** After accepting the proof, the Trust Manager uses ProofVerf() to verify if it is true or false. To persuade the data owner that their data is secure, the Trust Manager uses the audit results. Data blocks in file F are being challenged by the trust management system to ensure their integrity [11].

## 5. ALGORITHMS FOR SCDS-TM MODEL

This section discusses the proposed model's algorithms for owner, user, authentication server, and trust manager execution.

### Algorithms Invoked by the Owner

The KeyGen() function is used to generate an account's public, private, and master keys. Metadata is created as a consequence of preprocessing data file F. After obtaining the security parameter 1k, the owner sends the public key puk, the secret key sk, and the master key mk, all of which are encrypted using the master key. Index and signature are generated using input, the master key and secret key sk, and the File f [12]. The requested signature collection is delivered bi-on-bi. By signing the root R of a Distribution tree using SigSK(R), the owner ensures that it is authentic. Next, the index is compiled using the owner's input. In order to check the signature's authenticity, we utilise the public key puk, signature SigSK(D(R)), and a request "True" or "False". Upon successful verification, a signature SigSK(D(R)) with the value “True" is returned. It gives a "False" result if the old root R does not exist.

```
Algorithm1: Security Validation
Step1: Security parameter $1^k$
        Begin
            If ($1^k$)
        {
        (puk,sk,mk)□KeyGen($1^k$)
        return public key puk
        return secret key sk
        return master key mk
        }
        Step2: ECC Encryption
        Input: Message M, the curve base point BP, a random number r, owner’s secret key
        sk and a curve point P.
        Output: The cipher text C.
        Step2.1: Initially, the message M is mapped with curve point P.
        Step2.2: A random number is chosen such that r€ [1,n-1]
        Step2.3: The cipher text is obtained as, C ={(r*BP)+($P_m$+(r*sk))}
        Step3: Tag Generation
        Input: The cipher text C and the owner’s secret key sk
        Output: Tag
        Step3.1: Choose an element $y \leftarrow B$ and generates the tag T
        Step3.2: The tag includes name||n|| $y$ ||$SSig_{sk}$(name||n|| $y$ )
        Step4: Index Generation
        Input: Generates a binary data vector based on data file $F_i$.
        Output: sub-index τ
        Step4.1: The sub-index τ ={ $A_1^T D$ ,+ $A_2^T D$ „ }
        Step5: Signature Generation
        Input: Owner’s secret key sk, the File F
        Output: Signature Set S
        Step4: If R is the root
        then
        and signs it with private key α: $sig_{sk}$(DH(R) )← (DH(R) )$^\alpha$
        and Verify if{($sig_{sk}$) and m($sig_{sk}$(DH(R )),g)}=m(DH(R
        ),$g^\alpha$ then
            return True
```

```
    else
        return False
```

## 6. EXPERIMENTAL EVALUATIONS

Different measures and comparisons are made between the SCDS-TM proposal and other current frameworks, such as Erasure Coded, DPDP Scheme, and MRSE Techniques. We were able to better show the framework after experimenting with varied execution outcomes.

### 6.1 Lab Setup

The proposed SCDS-TM framework's security is compared to that of current systems including the erasure coded system, DPDP, and MRSE methods. The Java platform is used to implement the application [13]. An Intel two core CPU running 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Serial ATA hard drive are used to construct the user side. CloudSim software with greater immediate type, 7.5 GB memory, and 850 GB instant storage is used for the cloud server side.

### 6.2 Performance Analysis

- As well as providing data storage and retrieval, the SCDS-TM model decreases the owner's communication costs. Various indicators are used to assess the proposed SCDS-TM model's performance, including:
- **Communication Overhead**

The server's answer to the challenge is dealt with under the communication overhead section. Erasure coded schemes, DDPDP, and MRSE's communication overheads are examined in this section. KB/ms is the unit used to measure the communication overhead. The server response time is multiplied by the amount of data sent to determine the communication overhead (KB/ms) (ms) Theorem. Table 1 shows the communication overhead of the present SCDS-TM model and the proposed SCDS-TM model.

**Table 1: Comparison Chart for File Size vs Communication Overhead**

| File Size (KB) | Communication Overhead (KB/ms) | | | |
|---|---|---|---|---|
| | Existing Schemes | | | Proposed |
| | Erasure Coded Scheme | DPDP Scheme | MRSE Scheme | SCDS-TM Model |
| 50 | 0.35 | 0.34 | 0.30 | 0.28 |
| 100 | 0.45 | 0.46 | 0.44 | 0.43 |
| 150 | 0.42 | 0.43 | 0.40 | 0.35 |

| | | | | |
|---|---|---|---|---|
| 200 | 0.46 | 0.45 | 0.45 | 0.42 |
| 250 | 0.55 | 0.54 | 0.52 | 0.50 |
| 300 | 0.54 | 0.55 | 0.52 | 0.50 |
| 350 | 0.68 | 0.61 | 0.63 | 0.60 |
| 400 | 0.62 | 0.65 | 0.63 | 0.61 |
| 450 | 0.69 | 0.67 | 0.64 | 0.62 |
| 500 | 0.73 | 0.75 | 0.72 | 0.71 |

To see this in action, look at Fig. 4, where the SCDS-communication TM's overhead is much reduced since it doesn't need direct interaction from data owners. Multiple keywords are sent to the cloud server when using the MRSE approach, which increases the server's load and costs more money [14].

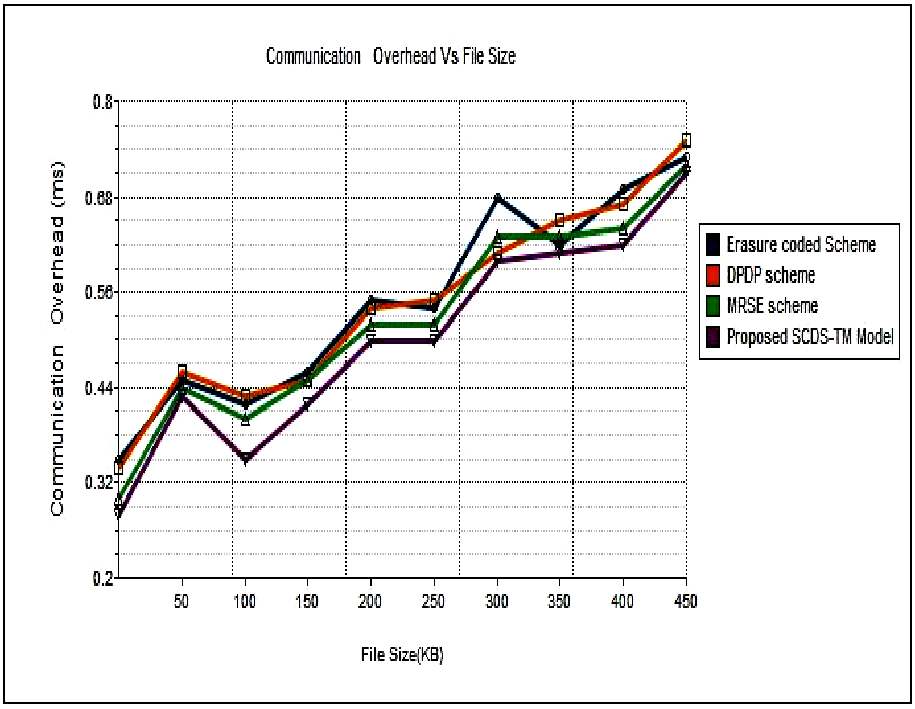

**Figure 4: Performance Graph of Block Size vs Communication**

ii. As a result, the suggested SCDS-TM model has a minimal communication overhead. Also, the suggested SCDS-TM model has a low communication overhead of 4% when compared to current systems like erasure coded, DPDP, and MRSE approaches. This is a significant improvement.

### iii. Data Transfer Rate

It's defined as the amount of time it takes to send data from a cloud server to a user. The SCDS-TM model's data transmission rate is compared to the rates of the erasure coded, DPDP, and MRSE methods. The unit of measurement is KB/ms.

$$\text{Data Transfer Rate } (KB/ms) = \frac{\text{Data Transferred by the Cloud Server to the User } (KB)}{\text{Time Taken } (ms)}$$

Table 2 shows a comparison of the SCDS-TM model's data transmission rate with the current methods. The proposed SCDS-TM model and current systems' data transmission rates are shown in Fig. 5. The cloud server retrieves the data requested by the user and conducts searching based on keywords and the index in the SCDS-TM model. As a result, only the required and specific information requested by the user is delivered to them, and the rest of the information is kept private. Because of this, the suggested SCDS-TM model has a 7 percent higher data transmission rate than other current methods.

**Table 2: Comparison Chart for File Size vs Data Transfer Rate**

| File Size (KB) | Data Transfer Rate (KBps) | | | |
|---|---|---|---|---|
| | Existing Schemes | | | Proposed |
| | Erasure Coded Scheme | DPDP Scheme | MRSE Scheme | SCDS-TM Model |
| **50** | 40 | 46 | 42 | 49 |
| **100** | 44 | 44 | 46 | 48 |
| **150** | 52 | 52 | 54 | 55 |
| **200** | 63 | 64 | 65 | 68 |
| **250** | 62 | 65 | 67 | 70 |
| **300** | 71 | 73 | 74 | 77 |

| | | | | |
|---|---|---|---|---|
| **350** | 74 | 72 | 75 | 81 |
| **400** | 88 | 83 | 82 | 88 |
| **450** | 92 | 91 | 95 | 105 |
| **500** | 112 | 117 | 121 | 125 |

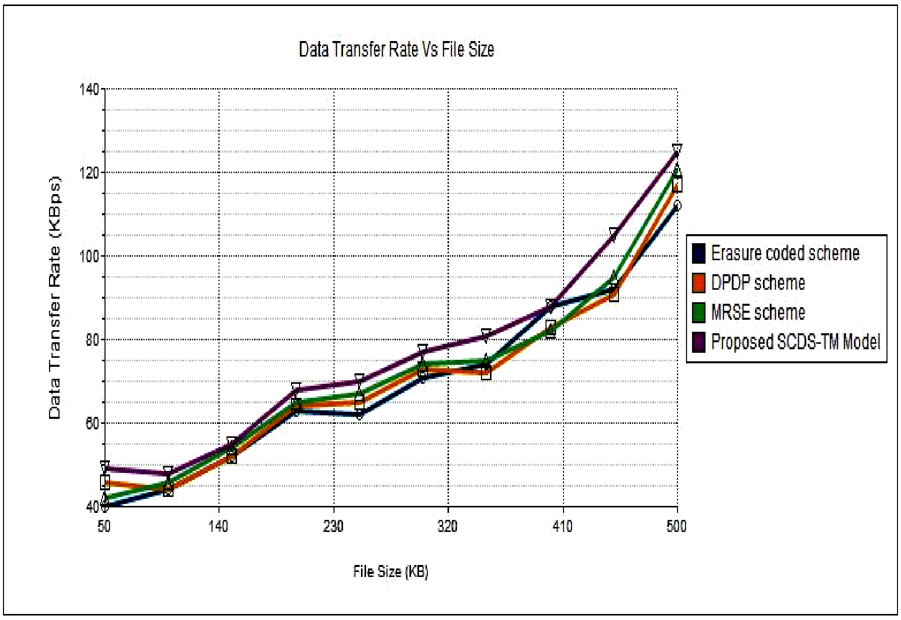

**Figure 5: Performance Graph of File Size vs Data Transfer Rate**

### iii. Query Execution Time

This gives an idea of how long it takes to carry out a certain query or request. When a user requests or queries the cloud server, the cloud server uses the keywords to search for relevant documents and the time it takes to run the proper query. A comparison of query execution time vs. block size for the proposed SCDS and DPDP systems is shown in the accompanying figure. Milliseconds are used to measure it. The following is the formula for calculating the execution time: time used by the user to send the query (ms) less the time taken by the server to reply (ms) Theorem (5.9) Table 3 shows the execution time of the SCDS-TM model and other current methods [15].

**Table 3: Comparison Chart for File Size vs Query Execution Time**

| **File Size** | **Query Execution Time(ms)** | | | |
|---|---|---|---|---|
| | **Existing Schemes** | | | **Proposed** |
| **(KB)** | **Erasure Coded Scheme** | **DPDP Scheme** | **MRSE Scheme** | **SCDS-TM Model** |

| | | | | |
|---|---|---|---|---|
| 50 | 10 | 10 | 6 | 5 |
| 100 | 11 | 12 | 10 | 10 |
| 150 | 13 | 15 | 11 | 11 |
| 200 | 15 | 18 | 14 | 12 |
| 250 | 18 | 20 | 15 | 14 |
| 300 | 24 | 26 | 22 | 20 |
| 350 | 28 | 26 | 25 | 24 |
| 400 | 35 | 36 | 33 | 31 |
| 450 | 44 | 45 | 43 | 41 |
| 500 | 49 | 46 | 45 | 43 |

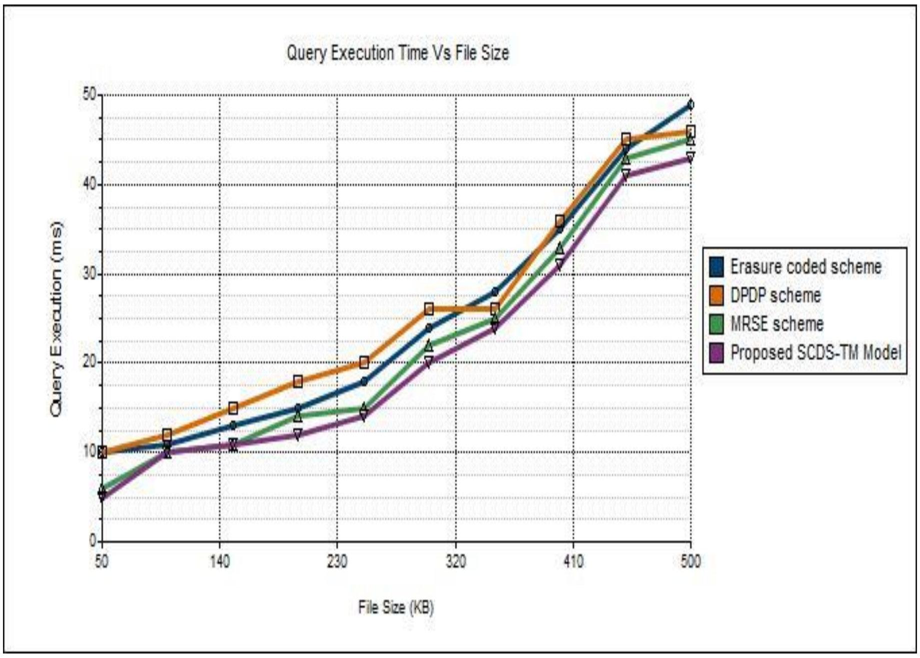

**Figure 6: Performance Graph of File Size vs Query Execution Time**

To summarise, the suggested SCDS-TM model's minimal query execution time is owing to its keyword-based search approach, as shown in Figure 6. The suggested SCDS-TM architecture has a 3 percent reduction in query execution time when the cloud server searches for the right keyword when the user requests the data.

### iv. Data Security Level

The quantity of cloud data that the server safely sends to the clients in relation to the total amount of data requested by the clients determines the data security level. In percentage terms, the degree of data security is determined ( percent ). The degree of data security is determined by

$$Data\ Security\ Level = \frac{Amount\ of\ Data\ Successfully\ Sent\ to\ the\ Clients}{Total\ Data\ Requested\ by\ the\ Clients} x\ 100$$

The data security level is computed and the performance analysis of the data security level based on different data in terms of KB is illustrated in Table 4.

**Table 4: Comparison Chart for File Size vs Data Security Level**

| File Size (KB) | Data Security Level (%) | | | |
|---|---|---|---|---|
| | Existing Schemes | | | Proposed SCDS-TM Model |
| | Erasure coded Scheme | DPDP scheme | MRSE Technique | |
| 10 | 55.35 | 54.34 | 57.22 | 61.34 |
| 20 | 57.68 | 55.13 | 58.29 | 63.74 |
| 30 | 58.20 | 56.49 | 58.30 | 64.55 |
| 40 | 59.05 | 57.28 | 62.37 | 65.45 |
| 50 | 62.83 | 60.21 | 63.71 | 66.42 |
| 60 | 64.26 | 62.55 | 65.68 | 67.22 |
| 70 | 66.39 | 63.27 | 69.10 | 70.42 |
| 80 | 67.02 | 65.72 | 70.24 | 72.11 |

| | | | | |
|---|---|---|---|---|
| 90 | 68.75 | 66.47 | 73.61 | 75.81 |
| 100 | 72.46 | 71.53 | 74.16 | 77.58 |

If you compare SCDS-TM to other current schemes of erasure coded, DDP and MRSE techniques, the data security level of SCDS-TM significantly improves by 5 percent shown in Fig. 7.

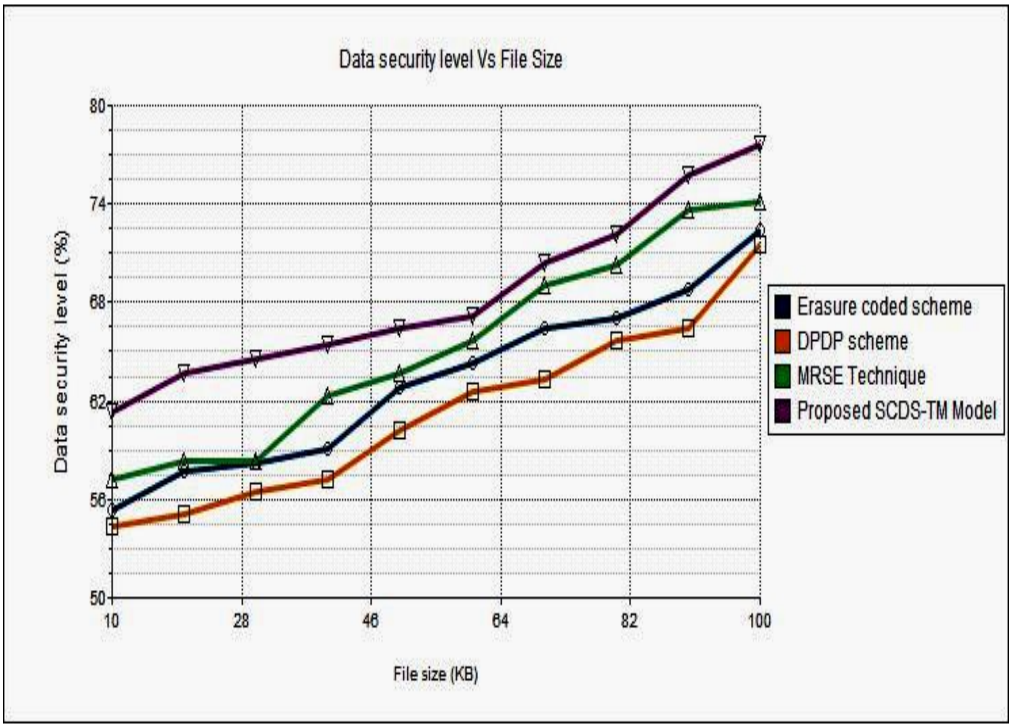

**Figure 7: Performance Graph for File Size vs Data Security Level**

### v. Probability Ratio

The act of successfully recovering the messages is what is meant by the probability ratio. Percentage is the unit of measurement. There are four primary components that determine the probability in an erasure coded approach, which include random selection of storage server and key server and random selection of random coefficients by storage and key server. The suggested SCDS-TM is evaluated in comparison to the other proposed SCDS-TM methods.

$$Probability\ Ratio\ (\%) = \frac{Number\ of\ \ Keywords\ in\ the\ Query\ Request}{Total\ Number\ of\ Keywords\ in\ the\ Server} \text{ x } 100$$

**Table 5: Comparison Chart for Number of Keywords vs Probability Ratio**

| Number of Keywords in | Probability Ratio (%) | |
|---|---|---|
| | Existing Schemes | Proposed |

| the Query/Request | Erasure Coded Scheme | DPDP Scheme | MRSE Technique | SCDS-TM Model |
|---|---|---|---|---|
| **10** | 50.15 | 53.11 | 55.63 | 57.23 |
| **20** | 52.25 | 54.26 | 56.27 | 59.54 |
| **30** | 61.21 | 63.56 | 64.37 | 65.48 |
| **40** | 62.32 | 63.41 | 65.04 | 66.15 |
| **50** | 65.29 | 64.34 | 66.10 | 68.38 |
| **60** | 72.23 | 73.34 | 74.47 | 75.01 |
| **70** | 74.52 | 75.61 | 76.59 | 77.20 |
| **80** | 78.53 | 75.54 | 80.52 | 81.29 |
| **90** | 82.43 | 79.41 | 83.43 | 84.63 |
| **100** | 83.29 | 80.81 | 84.26 | 85.15 |

In comparison to SCDM-TM, the probability ratio of obtaining the message in MRSE is medium as the probability ratio is bigger and therefore the correct message is recovered as the number of keywords in the request rises.

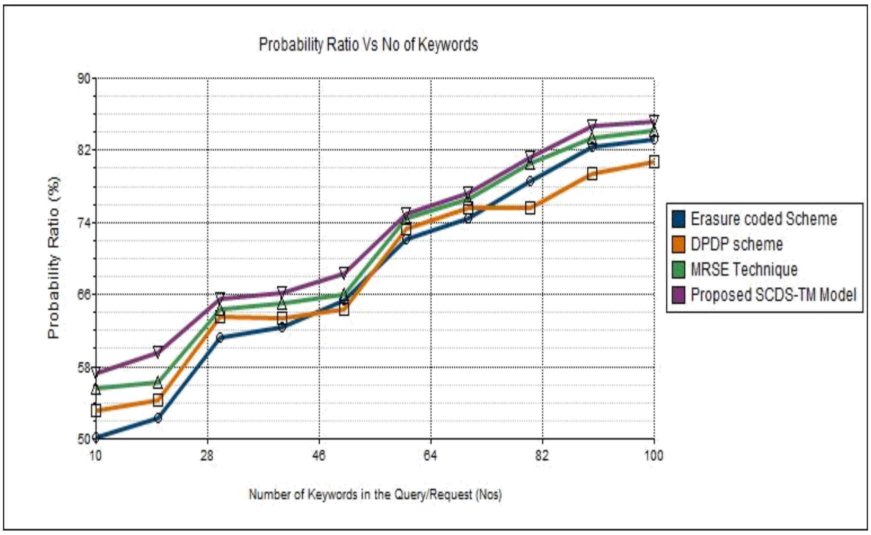

**Figure 8: Performance Graph of Number of Keywords vs Probability Ratio**

**vi.** When a user sends in a search query to retrieve a data file, the SCDS-TM model uses the index and tag to identify that data file, so the top –k documents relevant to that query are captured accurately using the coordinate matching technique, as shown in Fig. 9, and this provides a better probability ratio of 6% than that of the existing system.

### vii. Search Precision

It is common practise to introduce fake keywords into data vectors while computing similarity scores between papers. It's possible that some of the top k-documents will be omitted from the results because their true similarity ratings have fallen or that the similarity scores of other top k-documents have risen. The similarity ratings may not always be correct, thus a precision number is produced to assess the quality of the k documents that were successfully retrieved by you. The accuracy of a search is determined as follows:

$$\text{Precision Ratio (\%)} = \frac{\textit{Relevant Documents Retrieved by the User}}{\textit{Total Documents}} \text{x}100$$

As demonstrated in Table 6, existing schemes like DPDP, erasure coded, and MRSE were compared to the proposed SCDS-TM model and the results are provided.

**Table 6: Comparison Chart for Number of Retrieved Documents vs Precision Ratio**

| **Number of Retrieved Documents** | **Precision (%)** | | | |
|---|---|---|---|---|
| | **Existing Schemes** | | | **Proposed** |
| | **Erasure Coded Scheme** | **DPDP Scheme** | **MRSE Technique** | **SCDS-TM Model** |
| **50** | 35.17 | 38.23 | 41.19 | 43.11 |
| **100** | 42.46 | 45.42 | 52.15 | 55.31 |
| **150** | 53.29 | 54.46 | 56.37 | 58.40 |
| **200** | 60.43 | 61.47 | 63.82 | 65.44 |
| **250** | 66.70 | 65.52 | 67.28 | 69.75 |
| **300** | 71.20 | 70.02 | 72.65 | 73.33 |
| **350** | 72.31 | 73.82 | 74.33 | 75.28 |

| | | | | |
|---|---|---|---|---|
| **400** | 77.40 | 74.67 | 78.15 | 80.10 |
| **450** | 82.64 | 83.19 | 84.66 | 85.19 |
| **500** | 86.27 | 87.58 | 89.43 | 90.47 |

Because the search is dependent only on the search question, the accuracy of the MRSE technique's search results may be lacking. The suggested SCDS-TM model, on the other hand, uses the search query, index, and tag created for all documents in the file to improve search precision and, as a result, the precision ratio.

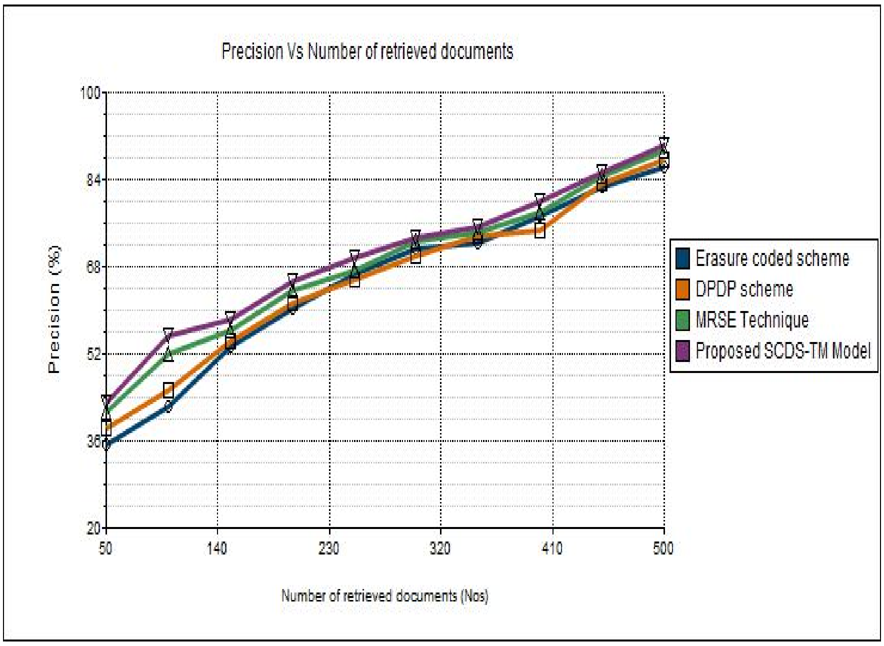

**Figure 9. Number of Retrieved Documents vs Precision**

As shown in Figure 9, the proposed SCDS-TM model has a higher accuracy ratio of 5% when compared to other existing systems. ECC file encryption, tag creation, signature generation, ECC decryption system, and index generation are all included in this paper's cloud storage security structure to enhance overall security and performance of the system. Additionally, four important issues are addressed in the proposed framework in order to enhance security.

Priority one should be given to maintaining the integrity and confidentiality of data, as well as enabling dynamic data operations and data retrieval. The security of consumer data is one of the most serious issues in a cloud-based system. Data security is therefore ensured via the Elliptic Curve Algorithm. To combat the risk of cloud-based data being tampered with, an integrity check has been included. The approach also allows for dynamic data manipulations.

Another problem is that the user may only access data relating to a certain phrase, and hence cannot access further sensitive information. Moreover, the proposed technique removes a step that has been skipped in all prior studies: the data owner does not have to make a search query or request directly to the cloud server.

Based on experiments, it is clear that the proposed solution surpasses existing techniques in terms of query execution time and communication overhead. By reducing query execution time and transmission costs while also increasing probability ratios, SCDS-TM is a triple win.

## 7. CONCLUSION

For users' data confidentiality, integrity, or retrieval, a secure SCDS-TM model has been developed for the cloud. Using the SCDS-TM paradigm, secret data is not only stored and retrieved securely; storage servers and key servers are also effectively coordinated. Dynamic data operations such as insertion or deletion of blocks may be performed using this model, which validates the integrity of the data blocks by using MHT construction. To make sure that data is complete and up-to-date, MHT employs a multi-step process. Added to that, the SCDS-TM framework's simplicity, efficiency, and strategy for public verifiability provide public verifiability without jeopardising the privacy of data owners. Data confidentiality and integrity must be seamlessly integrated into the protocol architecture in order to provide an elegant cloud storage system that enables efficient data storage, data integrity, and data retrieval. This SCDS-TM supports the distributed storage system, making it possible to store and retrieve data in a secure manner. A SCDS-TM model is offered here for quick data retrieval. Using an elliptic curve encryption method and coordinate matching, sensitive data may be recovered.

## REFERENCE

1. Hamlen, K. Kantarcioglu, M. Khan, L. Thuraisingham, B. Security Issues for Cloud Computing. International Journal of Information Security and Privacy, 4(2), 36-48. (2010).
2. Levina, N., and Vaast, E. The emergence of boundary spanning competence in practice: Implications for implementation and use of information systems. MIS Quarterly, 29(2), 335–363, (2005).
3. Ravishankar, M.N.; Pan, S.L.; and Leidner, D.E. Examining the strategic alignment and implementation success of a KMS: A subculture based multilevel analysis. Information Systems Research, 22(1), 39–59. (2011).
4. Tiwana, A, Novelty-knowledge alignment: A theory of design convergence in systems development. Journal of Management Information Systems, 29(1) 15– 52. (2012)
5. RizwanMian, Patrick Martin. Executing dataintensive workloads in a Cloud.ACM International Symposium on Cluster 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing.
6. Yingjie Shi, XiaofengMeng, Jing Zhao, Xiangmei Hu, Bingbing Liu and HaipingWang Benchmarking Cloud-based Data Management Systems.CloudDB'10, Toronto, Ontario, Canada. (2010).
7. Bernardo Ferreira, Henrique Domingos, Management and Search of Private Data on Storage Clouds.Center for Informatics and Information Technologies.SDMCMM'12, December 3-4, 2012.
8. XiaofengMeng, Adam Silberstein, Fusheng Wang, Information and Knowledge Management. CIKM'12, October 29–November 2, 2012

9. Peter Géczy, Noriaki Izumi, KôitiHasida Hybrid cloud management: Foundations and strategies. Review of business and finance studies. (4) 1, (2013).
10. Hussam Abu-Libdeh, Lonnie Princehouse, Hakim Weatherspoon (2010). RACS: A Case for Cloud Storage Diversity, ACM 978-1-4503-0036-0/10/06
11. Anthes, G. Security in the Cloud: Cloud Computing Offers Many Advantages, but Also Involves Security Risks. Communications of ACM, 53(11), 16-18. (2010).
12. Xiao-Bai Li, SumitSarkar Privacy Protection in Data Mining: A Perturbation Approach for Categorical Data Information Systems Research. (17) 3, 254–270, (2006).
13. Iyengar, V. S. Transforming data to satisfy privacy constraints. Knowledge Discovery DataMining. ACM Press, New York, 279–288. (2002).
14. Daniel J. Abadi (nd) Data Management in the Cloud: Limitations and Opportunities. IEEE Computer Society Technical Committee on Data Engineering
15. Gary Anthes, Security in the Cloud. Communications of the acm,(53) 11, (2010)