CHAPTER 4

SECURE COMMUNICATION CHANNEL FOR DATA SHARING IN CLOUD ENVIRONMENT

4.1. INTRODUCTION

Most of the organizations which would gain more benefits from cloud storage are stepping backwards because of the fear of their sensitive data being leaked out from cloud servers. The centralized data in cloud computing centers are processed with database and software application which manages the data. The users outsource their valuable data to the cloud servers to be maintained by the cloud service providers. This tends the service providers to increase the security and privacy on the outsourced data. Several schemes employing the security have been provided in cloud computing. However, most of them suffer from leakage of data. This leads to a bigger issue while sharing the data in the cloud computing environment.

In order to reduce the leakage of data in the cloud, an efficient technique named an Instant Communication Channel Key Organizer (ICCKO) model is constructed. It supports the security and data confidentiality in an efficient manner. The security of data provided by the ICCKO model is based on the communication between the client and the server and its performance and computational complexities are analyzed. Here, the proposed ICCKO model provides an end-to-end security in a highly distributed fashion. One of the modern features of the ICCKO model is its ability to maintain lightweight and powerful security. The proposed model provides secure data sharing with experienced data. It combines the aspects of access control for data sharing and usage control while accessing the data and authentication for maintaining the security. By means of the ICCKO model, the data vendor not only follows the service-level agreements that are being privileged, but also imposes access and usage control rules as needed.
4.2. ACCOUNTABILITY AND LEAKAGE DETECTION TECHNIQUES

Cloud computing offers an enormous computation power and storage capacity which facilitate the users to position their computation and data rigorous applications without infrastructure investment. However, preserving the data in cloud leads to a number of issues related to data loss, accountability, security etc. Such fears become a great barrier to the adoption of the cloud services by users. Hence, it becomes necessary for monitoring the usage of the client’s data in cloud.

4.2.1. Cloud Information Accountability Framework

The users must be able to confirm that the service level agreements made at the time of outsourcing is handled effectively in the cloud environment. The cloud Information Accountability makes use of JAR files for logging the usage of the client’s data by other users in the cloud domain [25]. The data are attached with access control policies and logging policies which the users want the cloud service provider to follow. The logging mechanism is triggered when the data is accessed by any user. By this, the data owner can have control on their own data irrespective of the location where it is stored. A link is maintained between the JAR and the data owner by keeping the JAR as the center point of contact. The loss of any log from the JAR is identified by recording the information regarding error correction sent by the JAR. If the center point is not able to be contacted by the JAR, the data enclosed in it cannot be accessed by the users.

4.2.1.1. Automated Logging Mechanism

Decentralized logging mechanism has to be used in cloud in order to get adapted to its nature. The data that needs to be controlled must be tightly bounded with the log files and an infrastructure support is necessary from the cloud server. Every data access has to be logged correctly by using the automated logging mechanism. The user who accesses the data has to be authenticated and verified. Moreover, the operations that have been executed on the client’s data as well as the time taken for execution have to be recorded by the logging mechanism. The log files must be secured in such a way that the malicious users are prevented from performing any insertion, deletion or modification in it. The log files which consist
of errors are restored by the recovery mechanisms. The log files must be provided in a regular manner to the data owner by the service provider to indicate them about the current usage of their data. The data owner also must be capable of retrieving the log files irrespective of the location where they are stored.

4.2.1.2. End-To-End Auditing Mechanism

The data usages are informed accurately to the user with the help of auditing mechanism. There are two types of auditing modes namely, push mode and pull mode. In the push mode, the harmonizer periodically pushes the logs to the data owner. Once the log files are sent to the owner, they are deleted to free the storage space which can be used for the next access logs. In addition to this, the information regarding error correction is also removed. Based on the log files obtained by the data owner, they are able to detect the entry of any malicious users using the checksum attached to every records. The pull mode makes the data owners to receive the logs from the cloud at any time when they are in need.

4.2.2. Data Leakage Detection

The sensitive data of the data owner are distributed to the authorized agents for their usage. But in some cases, the data can be leaked out and found in some unauthorized locations. The data leakage mechanism [32] detects whether the data is leaked out to unauthorized persons and if possible it identifies the agent who has leaked out the data.

4.3. AN INSTANT COMMUNICATION CHANNEL KEY ORGANIZER MODEL

The Instant Communication Channel Key Organizer (ICCKO) model in the cloud environment propose a data storage framework for the active data operations which is a missing factor in all the exiting schemes. In this section, the ICCKO model which provides efficient security in the cloud environment is described. The general framework of ICCKO model is illustrated in Fig.1.
Fig 4.1. General framework of cloud data storage

The objective of the ICCKO model is to support security and provide data confidentiality in an effective manner to the authorized clients. It is also a storage model for performing active data operations. The ICCKO model comprises of three different entities such as,

i) Client: The client consists of huge data files and largely depends on the cloud for computation and maintenance of data.

ii) Cloud Storage Servers (CSS): The cloud storage servers have enormous space to be used by different clients for storage purpose and computational resource for evaluating the client’s data.

iii) Belief Inspector: The Belief Inspector (BI) checks the authorization of the clients to access the data. The main objective of BI is to verify the integrity of data stored in cloud. It also delegates multiple tasks to different clients and further identifies the unauthorized clients.
4.3.1. General Architecture of the ICCKO Model

The cloud computing system consists of Cloud Service Provider (CSP), Belief Inspector (BI) and different types of clients. The Cloud Service Provider comprises of various Cloud Storage Servers (CSS). The CSS is a unit which is administered by the CSP. The clients request a portion of the data from the cloud storage servers. From the CSS, the CSP presents the flow of data to the BI for authorization purpose. The authorization is verified by the BI and accordingly the data is provided to the authorized clients. The architecture diagram of ICCKO model is shown in Fig 4.2.

![Architecture diagram of ICCKO model](image-url)
Fig 4.3. Data flow diagram of ICCKO model
In the ICCKO model, each communication is initiated with the channel assigned and provided with unique key by the organizer. The communication initiation for a channel helps the cloud data storage house to monitor the channel intruders and the appropriate sender and the receiver. The unique key for the communication model ensures the cloud security system to maintain communicational confidentiality across the cloud data access network. This model provides an end to end security with efficient data sharing to the authorized clients. The results shows that this approach further enhances the distribution of original data set and yields a disguised cloud with higher data security. The data flow diagram for the ICCKO model is given in Fig. 4.3.

4.3.2. Construction of ICCKO Model

To effectively support security without retrieving the data stream themselves, the instant communication model was constructed in which the metadata generated from individual data stream are securely combined. It uses the BI to verify the linear combination of data stream and checks whether it is correctly computed or not. In the following description, the ICCKO model is presented to illustrate the data sharing support in the cloud environment. The design of ICCKO model supports dynamic data operations which is one of the major challenging tasks for cloud storage systems.

4.3.2.1. Preliminaries

Some necessary cryptographic preliminaries for the proposed scheme is as follows:

- **Bilinear map**: The file $F$ is assumed to be divided into $m$ streams $n_1, n_2 \ldots n_m$ where $m_i \in X_p$, always $p$ is a large prime. Let $e: B^*B \rightarrow BT$ is a bilinear map, with a distributed function $D: \{0,1\}^* \rightarrow B$ viewed as random. Let $g$ be the generator of $B$ and $D$ that represents a cryptographic distribution function.

- **KeyGenr($1^k$)**: The client generates an systematic signing key pair (CPK, CSK). Choose an systematic $\alpha \leftarrow X_p$ and compute $w \leftarrow g^\alpha$. The secret key is $SK = (\alpha, CSK)$ and the public key is $PK = (w, CPK)$.

- **SignGen(SK, F)**: Given $F = n_1, n_2 \ldots n_m$ the client chooses an systematic
element w←B. Let q=name||m||w||CSig_{sk}(name||m||w) be the file mark for F. Then, the client computes signature $\sigma_i$ for each stream $n_i$ (i=1, 2,…,m) as $\sigma_i\leftarrow(D(n_i).u_n^\alpha)$. The client then generates a root R based on the construction of the Distribution Tree, where the leaf nodes of the tree are an ordered set of distribution of “file mark”. Next, the client signs the root R under the secret key $\alpha$:$\text{Sig}_{sk}(D(R))\leftarrow(D(R))^\alpha$.

4.3.2.2. An Instant Communication Channel Framework

The algorithm to describe the communication between the cloud storage servers and the client is provided here. It works on three modes, namely the client, server and belief inspector. The main idea behind the ICCKO model is presented below:

1. The file F is assumed to be divided into m streams $n_1$, $n_2$, … $n_m$, where $m_i \in X_p$ where, $p$ is a large prime.
2. The client generates a systematic key pair by invoking KeyGenr(1^k).
3. By executing SignGen(SK,F), the file F is preprocessed and generates a root R based on the construction of the Distribution Tree, and the instant communication channel together with metadata are produced.
4. Perform validation measure from the server using the function VerifyValidator (PK, validate, P_{validate}).
5. On successful verification, a signature $\text{SigSK}(D(R'))$ with a ‘True’ value for new root R’ is returned.
6. In case of verification being failure, a ‘False’ value is returned for the old root R.
7. The server takes as input the file F, its signature S and confront conf from the client and generates the proof P for data sharing.
8. The server gets the update request along with the file F, Signature ‘S’ from the client and generates updated file F’, Signature S’ and proof P_{validate} for the operation using ExecUpdate(F, S, Update) and finally returns P_{validate} to client.
9. The server sends the file to the BI if it is true using the function VerifyProof(PK,conf,P).
10. Upon receipt of the proof ‘P’, the authorization is done by performing a conditional checking, \((\text{SigSK (D(R))}, g) = ((D(R)), g^n)\).

11. If the proof verified is true, the BI sends the file to the client, otherwise the file is blocked from the client.

4.3.2.3. Client

The client’s public key and private key are generated by invoking KeyGenr() function. By running SignGen() function, the data file F is preprocessed, and the instant communication channel together with metadata are produced. The algorithmic representation of the functions done by the client are shown in Algorithm 4.1 as follows,

Algorithm 4.1 Functions of Client

//Security validation

**Input:** Security parameter as \(1^k\)

1. Begin
2. If \((1^k)\)
3. {
4. \((\text{PK, SK}) \leftarrow \text{KeyGenr}(1^k)\)
5. return public key PK,
6. return private key SK
7. }

//Signature generation

**Input:** Private Key SK and File F with ordered collection \(\{b_i\}\)

**Output:** Signature S set which is the ordered collection of signatures \(\{\beta_i\}\)

1. If (root R)
2. {
3. \((\text{SigSK (D(R))})\leftarrow \text{SignGen} (\text{SK}, F)\)
In security validation, initially the client takes an input security parameter $1^k$. The client returns the public key PK and secret key SK. Then for signature generation, the client takes as input, the private key SK and a file F with a collection of blocks $b_i$ and outputs the ordered collection of signature $\beta_i$ on $b_i$. The client also outputs the metadata in the form of signature $\text{SigSK(D(R))}$ of the root R of a Distribution tree. For verifying the validation, the client takes as input the public key PK, signature $\text{SigSK(D(R))}$ and a request ‘True’ or ‘False’ and validation measure from the server. On successful verification, a signature $\text{SigSK(D(R'))}$ with a ‘True’ value for new root R’ is returned. In case of verification being failure, a ‘False’ value is returned for the old root R.

4.3.2.4. Cloud Server

For generating the proof, the server takes input file ‘F’, its signature ‘S’ and confront ‘C’ from the client. The functions performed on the cloud storage server are represented in Algorithm 4.2 as,
Algorithm 4.2 Functions of Cloud Server

//Proof generator

Input: File ‘F’, Signature ‘S’ and Confront ‘C’
Output: Data sharing proof ‘P’ for the stream specified by C

1. If (C)
2. {
3. P ← GenProof (F, S, C)
4. Return stream of data
5. }

//Update execution

Input: File ‘F’, Signature ‘S’ and operation request ‘Update’ from client
Output: Updated File F’, Signature S’ and proof, P_{validate} for the operation

1. Update request from the client
2. If (File)
3. {
4. F’, S’, P_{validate} ← ExecUpdate(F, S, Update)
5. Return P_{validate} to client
6. }

The client sends the confront ‘C’ \{i, c_i\} to the server. Upon receiving the confront ‘C’ from the client, the server evaluates,

\[ \alpha = \sum_{i=1}^{n} c_i m_i \]

Equation (4.1)

where, \( \alpha \) represents the data sharing block,
\[ \sigma = \sum_{i=1}^{n} \sigma_i \in g \]  

Equation (4.2)

where, \( \sigma \) represents the signature block and outputs the proof \( P \) as specified by \( C \). For update execution, it takes as input file \( F \), signature \( S \) and data sharing operation request ‘Update’ from the client. It produces the new file \( F' \), new signature \( S' \) and proof \( P_{\text{validate}} \) from the client for further data sharing process.

4.3.2.4. Belief Inspector

The Belief Inspector (BI) checks the authorization of the clients for the purpose of providing data access to them. The algorithmic representation of the functions of BI is illustrated in Algorithm 4.3 as,

**Algorithm 4.3 Functions of Belief Inspector**

```plaintext
//Proof verifier

Input: Public Key PK, confront ‘C’ and Proof ‘P’ return by the server
Output: Returns file to appropriate client if it is true

1. Request Update from the client
2. (True, False) ← VerifyProof (PK, C, P)
3. If (Proof P)
4. { 
5. File sent to client
6. }
7. Else
8. { 
9. Client prevented from accessing the file
10. }
```

For verifying the proof, the BI takes as input the public key PK, confront C and proof P as returned by the server. Upon receipt of the proof P, the authentication
is done by performing a conditional checking, \((\text{SigSK}(D(R)), g) = ((D(R)), g^\alpha)\). If the proof verified is correct, it returns ‘true’ otherwise returns ‘false’. This algorithm is used to prove that a stored file is retrievable for the appropriate authenticated client.

4.4. EXPERIMENTAL EVALUATION

To compare the proposed Instant Communication Channel Key Organizer model with the existing Cloud Information Accountability (CIA) framework and Data Leakage Detection (DLD) technique, implementation is done in the Java platform. The cloud user side of ICCKO model is implemented on a workstation with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM, and a 7200 RPM Serial ATA drive. The cloud server side process is implemented on CloudSim software with larger instant type, 7.5 GB memory, and 850 GB instant storage. The randomly generated test data is of 1 GB size. The Amazon Access Samples dataset information is used on the transaction processing between cloud users and cloud servers.

The ICCKO scheme not only enables data sharing among multiple clients, but also reduces the computation cost on the Belief Inspector side. Given, total \(N\) clients in the system, the construction of ICCKO scheme helps to reduce the number of expensive combination operations. The performance of the proposed Instant Communication Channel Key Organizer model provides better communication with minimum disturbances in the channel. To evaluate the ICCKO model, the following metrics are evaluated which includes:

- Communication overhead
- Data transfer rate
- Data leakage detection rate
- Data security level
- Log creation time
- Channel disturbance level

Communication overhead refers to the overhead involved during the communication among the users. The instant communication channel performs better
communication between the users to extract the secure data from the cloud environment.

\[
\text{Communication Overhead} \left( \frac{\text{KB}}{\text{ms}} \right) = \frac{\text{Amount of data transferred (KB)} \times \text{delay for data transfer (ms)}}{}
\]

Equation (4.3)

Equation (4.3) shows that the communication overhead refers to extra bits transferred along with the original data which makes delay for the data to be transferred.

The data transfer rate is defined as transferring the data from one user to other users or between different users in a given time interval. It is measured in terms of KB/ms as,

\[
\text{Data transfer rate (KB/ms)} = \frac{\text{Data transferred (KB)}}{\text{Time taken (ms)}}
\]

Equation (4.4)

Equation (4.4) measures the total amount of data transferred and the time taken for its transfer through the communication channel.

Data leakage detection rate is the percentage of the data identified to be leaked to unauthorized users to that of total data lost. It is measured in terms of percentage (%).

\[
\text{Data leakage detection rate (\%)} = \frac{\text{Data identified to be leaked to unauthorized users (KB)}}{\text{Total data lost (KB)}} \times 100
\]

Equation (4.5)

Equation (4.5) measures the probability of data leakage detection capability of the belief inspector in identifying the data being leaked to unauthorized users.
The data security level is the amount of cloud data sent securely to the clients to that of total data requested. The data security level is measured in terms of percentage (%).

\[
\text{Data security level} = \frac{\text{Amount of data sent securely to clients}}{\text{Total data requested}}
\]

Equation (4.6)

Equation (4.6) measures the data security level provided to the data during its transfer to the clients.

The channel disturbance level is the level of disturbances incurred in the channel due to increased transfer of data. The channel disturbance level is measured as,

\[
\text{Channel disturbance level (\%)} = 100 - \text{Data fetched by user}
\]

Equation (4.7)

Equation (4.8)

From Equation (4.7) and (4.8), the level of disturbances occurred in the communication channel can be measured.

Log creation time is the time taken to create a log file when the entities are continuously accessing the data, causing regular logging for fetching the information in the ICCKO model. The log creation time is measured as,

\[
\text{Log creation time (ms)} = \log \text{in time for data access} - \log \text{out time after access}
\]

Equation (4.9)
Equation (4.9) deals with measuring the time taken for creating the log file. The log creation time is measured in terms of milliseconds (ms).

### 4.5. RESULTS AND DISCUSSION

To provide secure data sharing in the cloud storage server, the ICCKO model is analyzed and compared with existing Cloud Information Accountability (CIA) [25] and Data Leakage Detection (DLD) [32] schemes.

#### 4.5.1. Communication Overhead

Communication overhead occurs in a communication channel because of the delay that happens when large amount of data is sent through it. Table 4.1 shows the communication overhead in terms of delay occurred in transferring the data through the secure communication channel.

<table>
<thead>
<tr>
<th>Data file size (KB)</th>
<th>Communication overhead (KB/ms)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing schemes</td>
<td>Proposed ICCKO scheme</td>
</tr>
<tr>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>50</td>
<td>0.37</td>
<td>0.43</td>
</tr>
<tr>
<td>100</td>
<td>0.44</td>
<td>0.48</td>
</tr>
<tr>
<td>150</td>
<td>0.49</td>
<td>0.53</td>
</tr>
<tr>
<td>200</td>
<td>0.51</td>
<td>0.55</td>
</tr>
<tr>
<td>250</td>
<td>0.52</td>
<td>0.57</td>
</tr>
<tr>
<td>300</td>
<td>0.58</td>
<td>0.62</td>
</tr>
<tr>
<td>350</td>
<td>0.63</td>
<td>0.69</td>
</tr>
<tr>
<td>400</td>
<td>0.68</td>
<td>0.74</td>
</tr>
<tr>
<td>450</td>
<td>0.73</td>
<td>0.79</td>
</tr>
<tr>
<td>500</td>
<td>0.77</td>
<td>0.84</td>
</tr>
</tbody>
</table>
The communication overhead is measured for the ICCKO model and hence it is compared with that of existing CIA and DLD schemes as shown in Fig. 4.4. Since an instant communication channel is used for transferring the data to the clients, it is found that the ICCKO scheme offers reduced communication overhead of about 11-20% when compared to existing CIA and DLD schemes.

![Fig. 4.4. Data file size vs. communication overhead](image)

**4.5.2. Data Transfer Rate**

The data is transferred between the users and the cloud servers and the data transfer rate is measured which is shown in Table 4.2.

The data transfer rate is measured and comparison is made between the proposed ICCKO scheme with that of existing CIA and DLD schemes as shown in Fig. 4.5. It is found that the data transfer rate is increased to 10 – 16% in proposed ICCKO approach when compared to the existing CIA and DLD schemes.
Table 4.2. Tabulation for data transfer rate

<table>
<thead>
<tr>
<th>Data file size (KB)</th>
<th>Data transfer rate (KBps)</th>
<th>Existing schemes</th>
<th>Proposed ICCKO scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>43</td>
<td>40</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>56</td>
<td>50</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>95</td>
<td>85</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>155</td>
<td>135</td>
</tr>
<tr>
<td>250</td>
<td></td>
<td>220</td>
<td>206</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td>260</td>
<td>245</td>
</tr>
<tr>
<td>350</td>
<td></td>
<td>310</td>
<td>295</td>
</tr>
<tr>
<td>400</td>
<td></td>
<td>350</td>
<td>334</td>
</tr>
<tr>
<td>450</td>
<td></td>
<td>375</td>
<td>350</td>
</tr>
<tr>
<td>500</td>
<td></td>
<td>415</td>
<td>390</td>
</tr>
</tbody>
</table>
4.5.3. Data Leakage Detection Rate

The data leakage detection rate is measured by detecting the percentage of the data identified to be leaked during the transfer of data. Table 4.3 shows the data leakage detection rate of the proposed ICCKO scheme in comparison with the existing schemes like CIA and DLD.

Fig. 4.6 shows the data leakage detection rate of the ICCKO model and comparison is made with that of existing CIA and DLD schemes. In ICCKO scheme, each communication is instantiated with the channel assigned and provided with unique key by the organizer. The communication initiation for a channel helps the cloud data storage house to monitor the channel intruders and the appropriate sender and the receiver with the help of a BI. The BI identifies whether the data is leaked out to any unauthorized users. It is found that the data leakage detection rate is increased in proposed ICCKO approach to 8 – 15 % when compared to the existing CIA and DLD schemes.
Table 4.3. Tabulation for data leakage detection rate

<table>
<thead>
<tr>
<th>Total data lost (KB)</th>
<th>Data leakage detection rate (%)</th>
<th>Existing schemes</th>
<th>Proposed ICCKO scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>50</td>
<td></td>
<td>75.68</td>
<td>69.36</td>
</tr>
<tr>
<td>100</td>
<td></td>
<td>76.35</td>
<td>72.40</td>
</tr>
<tr>
<td>150</td>
<td></td>
<td>78.12</td>
<td>74.16</td>
</tr>
<tr>
<td>200</td>
<td></td>
<td>79.36</td>
<td>75.62</td>
</tr>
<tr>
<td>250</td>
<td></td>
<td>81.30</td>
<td>76.86</td>
</tr>
<tr>
<td>300</td>
<td></td>
<td>82.41</td>
<td>78.41</td>
</tr>
<tr>
<td>350</td>
<td></td>
<td>84.39</td>
<td>79.35</td>
</tr>
</tbody>
</table>

Fig 4.6. Total data lost vs. data leakage detection rate
4.5.4. Data Security Level

The data security level deals with the amount of cloud data transferred securely to the client. The data security level is measured in terms of percentage (%). The performance analysis of the data security level based on different data in terms of KB is illustrated in Table 4.4.

**Table 4.4 Tabulation for data security level**

<table>
<thead>
<tr>
<th>Data file size (KB)</th>
<th>Data security level (%)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing schemes</td>
<td>Proposed ICCKO scheme</td>
</tr>
<tr>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>50</td>
<td>68.36</td>
<td>70.22</td>
</tr>
<tr>
<td>100</td>
<td>69.11</td>
<td>72.36</td>
</tr>
<tr>
<td>150</td>
<td>70.10</td>
<td>73.69</td>
</tr>
<tr>
<td>200</td>
<td>71.44</td>
<td>75.87</td>
</tr>
<tr>
<td>250</td>
<td>72.32</td>
<td>78.24</td>
</tr>
<tr>
<td>300</td>
<td>73.85</td>
<td>79.36</td>
</tr>
<tr>
<td>350</td>
<td>74.10</td>
<td>80.25</td>
</tr>
<tr>
<td>400</td>
<td>75.89</td>
<td>81.36</td>
</tr>
<tr>
<td>450</td>
<td>76.36</td>
<td>82.44</td>
</tr>
<tr>
<td>500</td>
<td>77.36</td>
<td>83.89</td>
</tr>
</tbody>
</table>
The proposed ICCKO mechanism is compared with existing schemes such as CIA and DLD. Unique key for communication model ensures the cloud security system to maintain communicational confidentiality across the cloud data access network. The ICCKO model provides an end-to-end security with efficient data sharing to the authorized clients. From Fig. 4.7, the proposed ICCKO method improves the security level of data transaction to 13 – 17% when compared to the existing CIA and DLD schemes.

![Data file size vs. data security level](image)

**Fig. 4.7. Data file size vs. data security level**

### 4.5.5. Channel Disturbance Level

The disturbances that occur in the communication channel are measured and compared with the existing CIA and DLD approaches and is presented in Table 4.5. In the ICCKO method, the channel disturbances are minimized with the help of the belief inspector and provides flexible retrieval of data from the cloud server.
<table>
<thead>
<tr>
<th>Data requested by users (bytes)</th>
<th>Channel disturbance level (%)</th>
<th>Existing schemes</th>
<th>Proposed ICCKO scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>840</td>
<td>4.84</td>
<td>4.81</td>
<td>3.96</td>
</tr>
<tr>
<td>970</td>
<td>7.58</td>
<td>7.49</td>
<td>6.52</td>
</tr>
<tr>
<td>1200</td>
<td>7.92</td>
<td>7.86</td>
<td>6.34</td>
</tr>
<tr>
<td>1480</td>
<td>8.1</td>
<td>7.9</td>
<td>7.13</td>
</tr>
<tr>
<td>1600</td>
<td>8.24</td>
<td>8.21</td>
<td>7.75</td>
</tr>
<tr>
<td>1870</td>
<td>8.29</td>
<td>8.13</td>
<td>7.62</td>
</tr>
<tr>
<td>2100</td>
<td>8.96</td>
<td>8.61</td>
<td>7.54</td>
</tr>
<tr>
<td>2450</td>
<td>9.76</td>
<td>9.53</td>
<td>8.37</td>
</tr>
<tr>
<td>3000</td>
<td>10.59</td>
<td>10.23</td>
<td>10.21</td>
</tr>
</tbody>
</table>
The performance graph of the ICCKO approach for channel disturbance level is shown in the Fig 4.8. The ICCKO approach detects the unauthorized users who try to fetch data from the cloud server and are removed by the BI. This helps in reducing the channel disturbances to a great extent when compared to the existing methods. The channel disturbance level is found to be reduced to about 12-15 % when compared with CIA and DLD approaches.

4.5.5. Log Creation Time

Table 4.6 illustrates the log creation time measured based on the log file size. In the proposed ICCKO scheme, the time to create a log file increases linearly with the size of the log file. With this experiment as the initial point, the time is calculated as the difference between log in and log out time for accessing the data by the user.
<table>
<thead>
<tr>
<th>Log file size (KB)</th>
<th>Log creation time (ms)</th>
<th>Existing schemes</th>
<th>Proposed ICCKO scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CIA</td>
<td>DLD</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>200</td>
<td>10</td>
<td>11</td>
<td>6</td>
</tr>
<tr>
<td>300</td>
<td>12</td>
<td>15</td>
<td>8</td>
</tr>
<tr>
<td>400</td>
<td>16</td>
<td>18</td>
<td>13</td>
</tr>
<tr>
<td>500</td>
<td>20</td>
<td>22</td>
<td>15</td>
</tr>
<tr>
<td>600</td>
<td>23</td>
<td>25</td>
<td>18</td>
</tr>
<tr>
<td>700</td>
<td>25</td>
<td>28</td>
<td>20</td>
</tr>
<tr>
<td>800</td>
<td>28</td>
<td>32</td>
<td>23</td>
</tr>
<tr>
<td>900</td>
<td>31</td>
<td>36</td>
<td>24</td>
</tr>
<tr>
<td>1000</td>
<td>35</td>
<td>38</td>
<td>26</td>
</tr>
</tbody>
</table>

The log creation time of instant communication channel approach is illustrated in Fig 4.9. The time taken to create the log file is compared with the existing approaches. Comparison result shows that the log creation time is minimized in ICCKO scheme to 36 - 52 % when compared with the existing CIA and DLD schemes.
A novel framework of secure data sharing in the cloud storage server named as Instant Communication Channel Key Organizer model is introduced. The communication initiation for a channel helps the cloud data storage to monitor the channel intruders and the appropriate sender and the receiver by means of the belief inspector. Unique key for communication model ensures the cloud security system to maintain communicational confidentiality across the cloud data access network and reduces the data leakage that occurs in the channel. The experimental evaluations performed using Java language showed that Instant communication channel considerably outperforms especially in secure communication. Through the implementation, the results showed that the ICCKO scheme for the secure data sharing worked efficiently. The results are obtained with less channel disturbances and minimal communication overhead and thereby improves the data leakage detection rate.