An Instance Communication Channel Key Organizer Model for Enhancing Security in Cloud Computing

Thomas Brindha and Ramaswamy Shaji
Department of Information Technology, Noorul Islam University, India

Abstract: *Cloud computing has evolved as one of the next generation structural design of the IT endeavors in the recent years. It enables to function like the Internet through the process of accessing and sharing computing resources as virtual resources in a protected and scalable way, thereby obtaining huge influence in corporate data centers. Even as remotely hosted, governed services have long been a component of the IT landscape, an enhanced awareness in cloud computing has been powered by ubiquitous networks, maturing standards, the increase of hardware and software virtualization, and the drive to bring about IT costs flexible and obvious. Several schemes have been provided in cloud computing for enhancing the security, however most of them sustain from leakage of data. This leads to a greater issue while sharing the data in the cloud computing environment. In order to realize the leakage of data in the cloud, we address the construction of an efficient technique, Instance Communication Channel Key Organizer (ICCKO) model to support the security and data confidentiality in a secure manner. We formally prove the data security of ICCKO model based on client and server and analyze its performance and computational complexity. Performance of ICCKO model is evaluated in terms of communication overhead, log creation time, data transfer rate and channel disturbance level for sharing the data securely. Extensive security and performance analysis shows that the proposed scheme is tremendously efficient and provides with provable secure data sharing.*

Keywords: *Cloud computing, data sharing, security, key organizer, communication channel, data management.*

Received January 4, 2014; accepted May 10, 2014; published online October 29, 2015

1. Introduction

In the recent years, numerous trends are coming up in cloud computing, which is based on the internet development and use of computer technology. This is forever cheaper and additionally powerful processors together with the "software as a service" computing architecture are renovating data centers into pools of computing service on a massive scale. On the other hand, the rising network bandwidth and flexible network connections make it still possible for the clients to subscribe high-quality services from data and software that exist solely on isolated data centers.

Cloud computing offers an enormous computation power and storage capacity which facilitate the users to position computation and data rigorous applications without infrastructure investment. To decrease the re-computing cost, a huge amount of intermediary datasets can be produced and stored [14]. However, preserving the privacy of intermediate datasets becomes a challenging problem because adversaries might get better privacy-sensitive information by analyzing multiple intermediate datasets. Encrypting all datasets in cloud is extensively adopted in existing approaches to address this challenge.

Although, cloud acts as a hopeful service platform for the Internet, this innovative data storage standard in "Cloud" makes a lot of challenging issues which have reflective authority on the security and routine of the overall system. One of the major concerns with cloud data storage is that of data integrity confirmation at entrusted servers [15]. For example, the storage service provider which contains knowledge about the Byzantine failures sporadically might make a decision to cover the data errors from the consumers for the advantage of their own.

The most serious issue is that in order to save the money and to reduce the storage space, the service provider even neglect or delete the rarely accessed data files which belongs to an ordinary client. The major constraint is the large volume of outsourced data and the resource capacity of the client. To overcome the above problems, we propose a novel approach, namely Instance Communication Channel Key Organizer (ICCKO) model based on security. The security responsibility focuses on maintaining the usage of the data translucent and perceptible. Our proposed ICCKO framework provides end-to end security in a highly distributed fashion, since it has the ability to maintain lightweight and powerful security. ICCKO combines aspects of access control for data sharing, usage control while accessing the data and authentication for maintaining the security. The data vendors follow not only the service-level agreements that are being privileged, but also impose access and usage control rules as needed [3].

The rest of this paper is arranged as follows: Section 2 describes the research background and related works. Section 3 addresses the ICCKO model in cloud computing. Section 4 describes the construction of the ICCKO model. Section 5 shows the experimental evaluations. Section 6 shows the results and discussions of our experiment and finally, we conclude this paper in section 7.

2. Background and Related Works

Much of growing interest has been pursued in the context of security, the main issues which diminish the growth of cloud computing and impediment with data privacy and data protection persists to curse the market. With the advent of more information on individuals and companies provided in the cloud, the security aspect has to be addressed. A survey of different security issues for the service delivery models of a cloud computing system is provided in [10].

The particulars of the cloud computing security issues and to discover and create a secure channel to converse with the CSP while maintaining trust and confidentiality is provided in [1]. Virtualization can rise, by defending both the integrity of guest virtual machines and the cloud infrastructure mechanism. By defending the cloud infrastructure mechanism and the guest virtual machines integrity, the safety of cloud computing can be raised by virtualization. Particularly, the Advanced Cloud Protection System (ACPS) [6] is intended to provide improved cloud resources security.

The accuracy of the user's data can be ensured by means of an efficient and flexible distributed mechanism [12] with support to dynamic data operations [13]. The file distribution preparation is done based on error correcting code for ensuring the dependability of data [7]. When compared to the traditional file distribution techniques, this mechanism radically decreases the communication and storage overhead. From a security perspective, a number of unchartered risks and challenges have been introduced from this rearrangement to the clouds, worsening much of the efficiency of traditional protection mechanisms [16]. As a result, the security of the cloud is assessed by identifying the unique requirements of security and a feasible resolution eliminates these possible threats is presented. A distributed approach in which resources are allocated energetically to the users without investing in the infrastructure or licensing the software's on the client side is provided in [9].

A computing environment for cloud computing system is developed by delivering cross platform in to cloud computing system. A new cloud data processing framework is presented to process the sensitive information and to protect user's confidentiality in the cloud by storing data in a secure fashion. Cloud computing paradigm has elevated the heights of data storage to newer limits whereas decreasing capital expenditure. Cloud computing also includes risks connected to privacy, compliance regulations, identity, data security and the security of the network through which the data travels. Improved Broker Cloud Communication Paradigm (BCCP) [8] was provided for dealing with security issues. Three-tier architecture for securing the data is used to satisfy the different user's needs. Various access control rights can be specified for both sensitive and non-sensitive data. The main objective of data sharing is that a large volume of distributed data is shared among the network [2].

One of the main problems related to cloud computing is the data leakage by the agents. To improve the secure data sharing in the cloud computing environment and to minimize the data leakage, a new technique called ICCKO model is presented.

3. ICCKO Model in Cloud Computing

The ICCKO model in the cloud environment proposes a data storage framework for the active data operations which is a missing factor in all the exiting schemes. In this section, we present our ICCKO model, which extends the scalability and efficient security in cloud environment. The conceptual framework of ICCKO is illustrated in Figure 1. The objective of the ICCKO model is to support security and provide data confidentiality in a secured manner to the authorized clients. It is also, a storage model for active data operations.

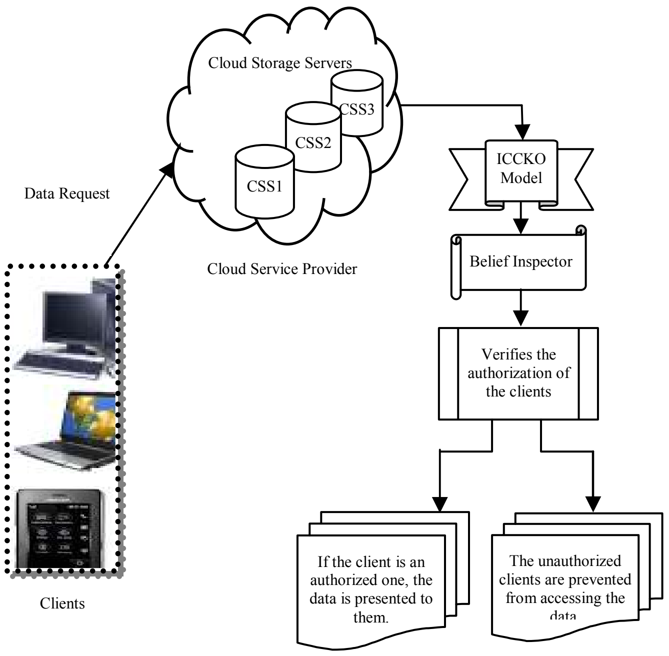
Cloud Storage Servers
CSS3
CSS2
CSS1
Cloud Service Provider
Data Request
ICCKO Model
Belief Inspector
Verifies the authorization of the clients
If the client is an authorized one, the data is presented to them.
The unauthorized clients are prevented from accessing the data
Clients

Figure 1. Architecture diagram of the proposed ICCKO model.

The ICCKO model comprises of three different entities. They are:

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

- Cloud Storage Servers (CSS): Have enormous space to be used by different clients for storage purpose and computational resource for evaluating the client's data.
- Belief Inspector (BI): Finally, the BI checks for further authorization of clients to access the data or not. The main objective of BI is to verify the integrity of data stored in cloud. It also, delegates multiple tasks from different client and further checks for authorized clients. In addition, BI also, minimizes data leakage by providing authorization to the clients to access sensitive data by sharing the information only with the authorized users.

As depicted in Figure 1, the cloud computing system consists of Cloud Service Provider (CSP), BI and different types of clients. The CSP comprises of various CSS. The CSS is a unit which is administered by the CSP. The client request the portion of data from the CSS which are administered by the CSP in a regular manner. From the CSS, the data flow is presented to the BI for authorization purpose. In the ICCKO model, each communication is instantiated with the channel assigned and provided with unique key by the organizer.

The communication instantiation for a channel helps the cloud data storage house to monitor the channel intruders and the appropriate sender and the receiver. Unique key for 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 result shows that this approach further enhances the distribution of original data set and yields a disguised cloud with higher data security.

The overall workflow diagram of the ICCKO model is shown in Figure 2.

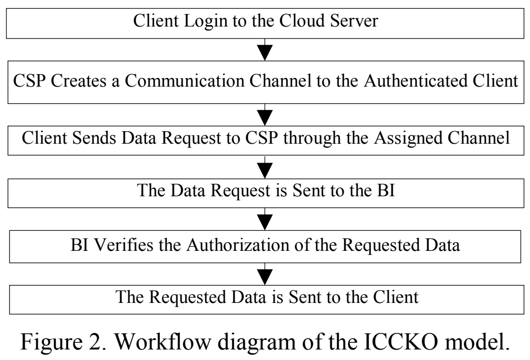
Client Login to the Cloud Server
CSP Creates a Communication Channel to the Authenticated Client
Client Sends Data Request to CSP through the Assigned Channel
The Data Request is Sent to the BI
BI Verifies the Authorization of the Requested Data
The Requested Data is Sent to the Client

Figure 2. Workflow diagram of the ICCKO model.

4. Construction of ICCKO Model

To effectively support security without retrieving the data stream themselves, we resort to the instance communication model in which the metadata generated from individual data stream is securely combined. It ensembles with the BI which verifies the linear combination of data stream and checks whether it is computed correctly or not. In the following description, we present the ICCKO scheme to illustrate data sharing support. To overcome the security problems, ICCKO protocol design for supporting dynamic data operations is introduced which is one of the major challenging tasks for cloud storage systems.

4.1. Preliminaries

Some necessary cryptographic background [4, 5] for our proposed scheme is as follows:

- Bilinear Map: We assume that file F which is divided into m streams n_1, n_2, ..., n_m where m_i ™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}*→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 unsystematic signing key pair (CPK, CSK). Choose an unsystematic $\alpha\leftarrow X_p$ and compute $w\leftarrow g^{\alpha}$. The secret key is SK= (α, CSK) and the public key is PK=(w, CPK).
- SignGen(SK, F): Given F=n_1, n_2 , ..., n_m the client chooses an unsystematic element $w\leftarrow B$. Let $q=name||m||w||CSig_{csk}(name||m||w)$ be the file mark for F. Then, the client computes signature σ_i for each stream n_i(i=1, 2, …, m) as $\sigma_i\leftarrow(D(n_i).u^{ni})^{\alpha}$. 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 $\alpha{:}Sig_{sk}(D(R))\leftarrow(D(R))^{\alpha}$.

4.2. Instance Communication Channel Framework

We formally present the algorithm to describe the communication between the CSS and the client. The algorithm works on three modes, namely:

1. The Client.
2. The Server.
3. The BI.

The main idea behind the ICCKO model is presented below:

1. We assume that file F is divided into m streams n_1, n_2, ..., n_m, where $m_i \in X_p$, 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 instance 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 $SigSK(D(R'))$ 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 confronts 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, $(SigSK(D(R)), g)=((D(R)), g^{\alpha})$.
11. If the proof verified is true, the BI sends the file to the client, otherwise the file is blocked from the client.

4.2.1. Client

The client's public key and private key are generated by invoking KeyGenr() function. By running SignGen() function, the data file F is pre-processed and the instance communication channel together with metadata are produced.

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 bi and outputs the ordered collection of signature β_i on b_i.

The client also outputs the metadata in the form of signature *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 *SigSK*(*D*(*R*)) and a request 'True' or 'False' and validation measure from the server. On successful verification, a signature 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*. The step by step process run by the client is provided in Algorithm 1.

Algorithm 1: Client.

```
Step 1: Security Validation.
Input: Security parameter as $1^k$

Begin
If ($1^k$)
   {
      (PK, SK) ← KeyGenr($1^k$)
      return public key PK
      return private key SK
   }
Step 2: Signature Generation
Input: Private Key SK and File F with ordered collection {bi}
Output: Signature S set which is the ordered collection of signatures{$\beta_i$}

If (root R)
   {
      (SigSK (D(R)))←SignGen(SK, F)
   }
Step 3: Verification Validator
Input: Public Key PK and signature SigSK(D(R))
Output: Signature SigSK(D(R')) for new Root R' and value true or false

Request "True"
   {
      (True, False, SigSK(D(R')))←
      VerifyValidator(PK, validate, $P_{validate}$)
      Return True
   }
Else
   {
      Return False
   }
```

4.2.2. Cloud Server

For generating the proof, the server takes as input file F, its signature S and confront conf from the client. The client sends the confront conf $\{i, c_i\}$ to the server. Upon receiving the confront conf from the client, the server evaluates:

$$\alpha = \sum_{i=1}^{n} c_i m_i \tag{1}$$

Where α represents the data sharing block.

$$\sigma = \sum_{i=1}^{n} \sigma_i \in g \tag{2}$$

Where σ represents the signature block and outputs the proof P as specified by conf. 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 confront from the client for further data sharing process. The functions done by the cloud server is provided in Algorithm 2.

Algorithm 2: Cloud server.

```
Step 1: Proof Generator
Input: File F, Signature 'S' and Confront 'conf'
Output: Data sharing proof P for the stream specified by conf

 If (conf)
   {
      P← GenProof (F, S, conf)
      Return stream of data
   }
Step 2: 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

Request Update from the client
 If (File)
   {
      F', S', $P_{validate}$←ExecUpdate(F, S, Update)
      Return $P_{validate}$ to client
   }
```

4.2.3. Belief Inspector

For verifying the proof, the BI takes as input the public key PK, confront conf and proof P as returned by the server. Upon receipt of the proof P, the authentication is performed by performing a conditional checking, $(SigSK(D(R)), g) = ((D(R)), g^{\alpha})$. If the proof verified is correct, it returns 'true' otherwise returns 'false'. The algorithm is used to permit a set of servers to prove that a stored file is intact and retrievable for the entire appropriate authenticated client. Algorithm 3 represents the functions done by the BI.

Algorithm 3: Belief inspector.

```
Step 1: Proof Verifier
Input: Public Key PK, confront 'conf' and Proof 'P' returned by the server
Output: Returns file to appropriate client if it is true

Request Update from the client
(True, False)← VerifyProof (PK, conf, P)
If (Proof P)
  {
    File sent to client
  }
Else
  {
    Client prevented from accessing the file
  }
```

5. Experimental Evaluations

To compare ICCKO model with the Cloud Information Accountability (CIA) Framework, implementation is done in the Java platform. In our experiment, the *BI* /user side process is implemented on a workstation with an Intel Core 2 processor running at 1.86GHz, 2048MB of RAM and a 7200 RPM Serial ATA drive. The cloud server side process is implemented on CloudSim 3.0 with larger instance type, 7.5GB memory and 850GB instance storage. The randomly generated test data is of 1GB size.

We also, conducted experiments for multiple clients and demonstrated its efficiency where the number of clients in the system is increased from 1 to approximately 100 with intervals. As we can see, ICCKO not only enables data sharing between the multiple clients, but also reduces the computation cost on the BI side. Given total *N* clients in the system, the construction of ICCKO model helps to reduce the number of expensive combination operations. The performance of the proposed model provides the better communication with minimum disturbances in the channel which is measured in terms of:

- Communication Overhead.
- Log Creation Time.
- Data Transfer Rate.
- Channel Disturbance Level.

6. Results and Discussion

The evaluation is done for different instance sizes, by varying the n number of servers. For each value of n, the minimum count of 250 instances was generated. For each algorithm, the line connects the average values, the upper error bar shows the maximum number of instances and the lower error bar shows the least value of instances. The tables and graphs given below shows the performance of the proposed IC CKO model compared with an existing CIA framework [11].

6.1. Communication Overhead

The instance communication channel performs better communication between the users to extract the secure data from the cloud environment. The communication overhead in the proposed technique is less when compared with the CIA approach.

Table 1 describes the chance of communication overhead raise based on the number of user participation. The outcome of the ICCKO is compared with the CIA techniques. Figure 3 describes the communication overhead based on the number of instances in it. The communication overhead is measured in terms of loss of data met when more number of clients arrived into the environment. The proposed model follows the key procedure in which the fetching of the data from the server further reduces the overhead while performing the communication.

Table 1. No. of instances vs. communication overhead.

No. of Instances	**Communication Overhead**	
	CIA Technique [11]	**Proposed ICCKO**
250	38	13
500	40	19
750	44	24
1000	48	28
1250	53	34
1500	55	37
1750	59	39
2000	63	43
2250	70	45

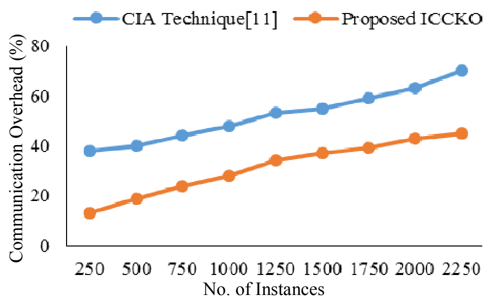

Figure 3. Data attributes vs. communication overhead.

Compared to the CIA approach which concerns only about the information responsibility, the proposed ICCKO takes into consideration both data sharing and security to process the user's tasks with minimum communication overhead. This in turn improves the probability of identifying the data leakage by the BI. The communication overhead is approximately 25-30% low in the proposed ICCKO model.

6.2. Log Creation Time

The time taken to create a log file when the entities are accessing the data causes continuous logging for fetching the information in the ICCKO model. Table 2 describes the log creation time with 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.

Table 2. Log file size KB vs. log creation time.

Log File Size (KB)	Log Creation Time (ms)	
	CIA Technique [11]	Proposed ICCKO
100	10	2
200	20	4
300	25	5
400	30	6
500	31	6
600	40	8
700	45	9
800	50	10
900	60	12
1000	68	13

The log creation time of instance communication channel approach is usually 5 times faster than the CIA approach. Comparison result of ICCKO model with an existing CIA approach based on log creation time consumption variance is measured in terms of milliseconds (m/s). The performance graph of the ICCKO model for creating log time is shown in the Figure 4. The variance in the execution time consumption would be 15-20% low in the ICCKO model.

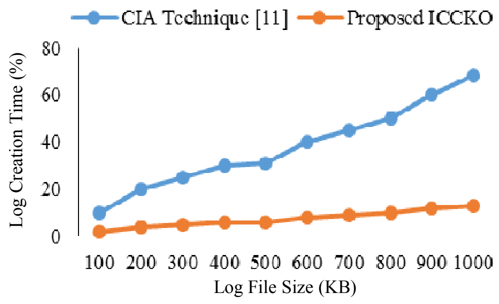

Figure 4. Log file size (KB) vs. log creation time.

6.3. Data Transfer Rate

The ICCKO model performs a communication channel for the efficient data transferring with the help of the key size. Table 3 describes the data transfer from the client to the cloud storage server based on the key size in it. The outcome of the proposed ICCKO is compared with the CIA approach.

Table 3. Key size vs. data transfer rate.

Key Size (KB)	Data Transfer Rate (%)	
	CIA Technique [11]	Proposed ICCKO
1000	70	98
2000	75	97
3000	76	96
4000	78	95
5000	79	95
6000	80	92
7000	79	93
8000	81	91
9000	82	90
10000	82	90

Figure 5 shows the experimental results on the key size, with respect to the data transfer set. Our results show that the data transfer rate is higher when compared to the CIA approach as the unauthorized users are identified by the BI responsible for data leakage and presented with only the data to the authorized clients neglecting the unauthorized clients. Both the key size, i.e., the private key and public key size are smaller in size to reduce the memory consumption for storing the data in the cloud servers.

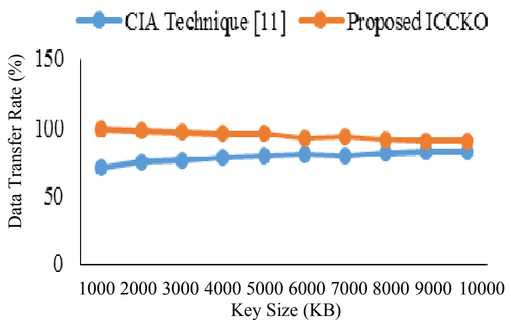

Figure 5. Key size vs. data transfer rate

The data transfer rate is measured in terms of kilobytes per second (kbps). Compared to the existing CIA approach, the ICCKO model for cloud computing performs a reliable data transfer rate and the variance is approximately 20-25% high in the ICCKO model.

6.4. Channel Disturbance Level

The channel disturbance level is the level of disturbance occurs when the communication take place from the client to the cloud environment in the ICCKO model.

Table 4 describes the occurrence of channel disturbance during communication. In the ICCKO model, the procedure of disturbance is restricted with the help of the BI and easily fetches the data from the cloud with no disturbance in the channel. The channel disturbance in the proposed scheme is less when compared with the CIA approach. The comparison result of ICCKO model with an existing CIA approach based on channel disturbance variance is measured in terms of percentage (%).

Table 4. No. of users vs. channel disturbance level.

No. of Users	Channel Disturbance Level (%)	
	CIA Technique [11]	Proposed ICCKO
500	10	2
1000	15	3
1500	16	5
2000	18	6
2500	18	7
3000	20	8
3500	22	8
4000	23	8
4500	24	9

The performance graph of the proposed ICCKO model for channel disturbance level is shown in Figure 6. This approach with key organizer identifies the unauthorized clients who try to fetch data from the client and are removed further by BI. This approach between the BI and clients reduces channel disturbance when compared to CIA technique. The variance in the channel disturbance level would be approximately 10-15% low in the ICCKO model.

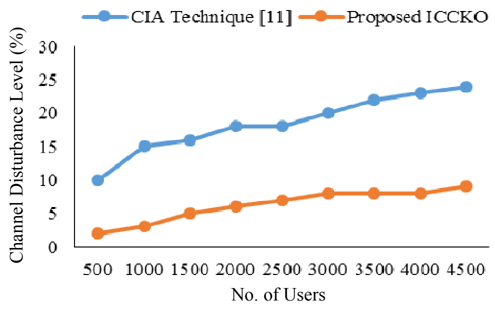

Figure 6. No. of users vs. channel disturbance level.

Figure 7 shows a screen shot of the client's data request to CSP and Figure 8 shows the screenshot of data retrieval from the server.

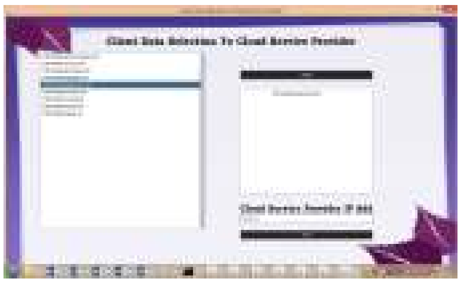

Figure 7. Data request to CSP.

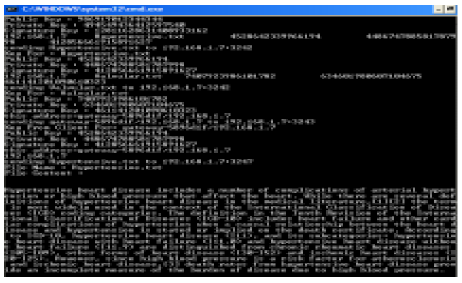

Figure 8. Data retrieval from CSP.

7. Conclusions

In this work, we have proposed a novel framework of secure data sharing in the cloud storage server by proficiently introducing the ICCKO model. The basic approach for identifying and minimizing the data leakage through BI is presented which reduces the probability of data leakage in the system. The framework addresses the unique challenges brought by ICCKO by enhancing the data sharing compared with the previous works. To achieve efficient data sharing, we improve the existing proof of storage models by manipulating ICCKO model. Another major concern is how to provide security for the data in the cloud environment. In this work, we explored the key organizer model to support the security and provide data confidentiality. To support efficient handling of multiple delegated tasks, we further explore the mechanism of bilinear map. We compared ICCKO with CIA framework in terms of log creation time to create and share the data securely. Our construction is intentionally designed to meet these two important goals while efficiency being kept intimately in mind. The experimental evaluations showed that the Instance communication channel considerably outperforms CIA especially in secure communication. Through implementation results showed that the ICCKO scheme for secure data sharing worked efficiently with less channel disturbance and minimal communication cost.

References

[1] Ahmed M., Xiang Y., and Ali S., "Above the Trust and Security in Cloud Computing: A Notion towards Innovation," in *Proceedings of the 8th International Conference on Embedded and Ubiquitous Computing,* Hong Kong, pp. 723-730, 2010.

[2] Gannouni S., Beraka M., and Mathkour H., "A Critical Comparison for Data Sharing Approaches," *The International Arab Journal of Information Technology*, vol. 11, no. 1, pp. 69-77, 2014.

[3] Li M., Yu S., Zheng Y., Ren K., and Lou W., "Scalable and Secure Sharing of Personal Health Records in Cloud Computing using Attribute-based Encryption," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131-143, 2013.

[4] Liu H., Zhang P., and Liu J., "Public Data Integrity Verification for Secure Cloud Storage," *Journal of Networks*, vol. 8, no. 2, pp. 373-380, 2013.

[5] Liu X., Zhang Y., Wang B., and Yan J., "Mona: Secure Multi-owner Data Sharing for Dynamic Groups in the Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1182-1191, 2013.

[6] Lombardi F. and Pietro R., "Secure Virtualization for Cloud Computing," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1113-1122, 2011.

[7] Madhavi K., Tamilkodi R., and Baladinakar R, "Data Storage Security in Cloud Computing for Ensuring Effective and Flexible Distributed System," *International Journal of Electronics Communication and Computer Engineering*, vol. 3, no. 1, pp. 133-137, 2012.

[8] Raj G. and Kaur K., "Secure Cloud Communication for Effective Cost Management System through MSBE," *International Journal on Cloud Computing: Services and Architecture*, vol. 2, no. 3, pp. 19-30, 2012.

[9] Singh G., Sharma A., and Lehal M., "Security Apprehensions in Different Regions of Cloud Captions Grounds," *International Journal of Network Security and Its Applications*, vol. 3, no. 4, pp. 48-57, 2011.

[10] Subashini S. and Kavitha V., "A Survey on Security Issues in Service Delivery Models of Cloud Computing," *Journal of Network and*

Computer Applications, vol. 34, no. 1, pp. 1-11, 2011.

[11] Sundareswaran S., Squicciarini A., and Lin D., "Ensuring Distributed Accountability for Data Sharing in the Cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 4, pp. 1555-5671, 2012.

[12] Wang C., Chow S., Wang Q., Ren K., and Lou W., "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362-375, 2013.

[13] Yang K. and Jia X., "An Efficient and Secure Dynamic Auditing Protocol for Data Storage in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 9, pp. 1717- 1726, 2013.

[14] Zhang X., Liu C., Nepal S., Pandey S., and Chen J., "A Privacy Leakage Upper-Bound Constraint Based Approach for Cost-Effective Privacy Preserving of Intermediate Datasets in Cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1192-1202, 2013.

[15] Zhu Y., Ahn G., Hu H., Yau S., An H., and Chen S., "Dynamic Audit Services for Outsourced Storages in Clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 2, pp. 227-238, 2013.

[16] Zissis D. and Lekkas D., "Addressing Cloud Computing Security Issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp.583-592, 2012.

Thomas Brindha received her ME degree in Software Engineering from Periyar Manniammai College of Engineering, Anna University, India and pursuing her PhD degree in Computer Science and Engineering at Noorul Islam University. Currently, she is working as an Assistant Professor in the Department of Information Technology at Noorul Islam University. Her area of interest is service-oriented computing, parallel processing, distributed computing and cloud computing.

Ramaswamy Shaji received his MTech degree in Computer Science and Engineering at Pondicherry, India and his PhD degree in Computer Science and Engineering from Manonmaniam Sundaranar University, Tirunelveli, India. Currently, he is working as a Professor in the Department of Information Technology at Noorul Islam University. His current research interests include pervasive computing and QoS in heterogeneous networks.