ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

Maintenance of Entropy through Token Parcelling and Verification of Error Bumping Off For Protection in Cloud Computing

Bhavya Boggavarapu[1], Baby sindhura Katragadda[2], V. P. Krishna Anne[3]

Department of CSE, K L University, Andhra Pradesh, India[1]
Department of ECE, K L University, Andhra Pradesh, India[2]
Department of CSE, K L University, Andhra Pradesh, India[3]

Abstract—***Secure Maintenance of data implementation through a large cloud, which is considered to be as an internet database, which is dynamic in fashion. It involves all the entropy values which are to be saved as to access it and it is the main threat removing scope in the present scenario which can play a key role in helping the organisation to withstand the effect it is facing from any personnel usage of activities further. As it is envisioned as next generation architecture of IT endeavour. This paper addresses the important aspect of providing quality of service and storage security. To make sure the correctness of users' data in the cloud which is proposed effectively and flexibility using decentralized scheme opposing to its predecessors by utilizing the similar token with distributed verification of erasure-coded data. Thus proposed scheme achieves the integration of storage, correctness insurance, and data error localization i.e., identification of misbehaving servers including modifications like deletion, insertion, updating and append. Analysis shows that proposed scheme is highly efficient and resilient against tangled failures, malicious data modification attacks.***

I. INTRODUCTION

Cloud Computing enables to open up the era in several trends. As it is an internet based deployment and computerized technology. Cloud Computing uses the ever cheaper and more powerful processors together with the complete outsourcing arena involving software as a service (Saas) , Internet as a Service (Iaas), (Paas) with these computer architecture for transforming data centres into pools of computing service on a huge scale. The simplest thing that a computer does is to store in the allocated space and retrieve information whenever requested by the authenticated user. We can store any kind of data that is we use in our daily panorama to endues confidentiality. The growth in the network bandwidth and reliable yet flexible interlinked connections make it even implementable that users can get started with high quality services from data and software that reside on remote entropy locations. Passing on information into the cloud architecture offers great viability to users since, they need not care about the complications involved in direct hardware management. Some of the major tautens like – Amazon, Microsoft and Google have implemented the "CLOUD" to speed up their by play.

The cloud helps endeavours to have a dynamically scalable abstracted reckoning infrastructure that is available on – demand. This model not only saves the IT teams from investing but also protects them from involutions in infrastructure setup and maintenance. Apart from providing the on-demand infrastructure, cloud service provides by and large interfaces for the other related IT management services.

From the perspective of data security which has always been an important aspect of quality of service inevitably poses new challenging security threats for number of reasons. Initially traditional cryptographic perspectives for the purpose of ensuring data protection cannot be directly adopted due to users' loss control. In order to minimize this loss verification of correct data storage in the cloud must be conducted without explicit glance of the complete data. Considering the various kinds of entropy for each user stored in the cloud and the demand of long term on going insurance of their data safety, the problem of verifying correctness of stored data becomes even more challenging in the cloud as it is not just a third party data warehouse to store data without providing any security tasks.

The database operations involve frequent insertion, deletion, modification, appending, reordering and updated by the users' etc.. Hence, it can be proved as dynamic feature which makes traditional integrity insurance perspectives futile and entails new solutions. There is also a scope to minimize problem causing aspects like –

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

- Data Redundancy
- Data uncertainty
- Data Without Validation
- Data Without Confidentiality

Performed to the respective person headed-up with it, without any end in multiple physical locations to further calm down the data integrity threats.

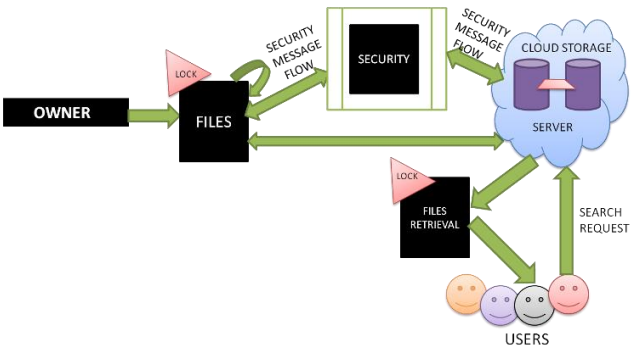
As we know that "Cloud Computing" is a web-Based applications that change the way you work and collaborate online to ensure storage correctness, wellness without having users possessing data. This is conquered using decentralized protocols for ensuring storage correctness across numerous servers or peers. In order to prevent the above listed problems we need to maintain tokens to the allocated data which need to refer by themselves whether there exist any simulation or not. Through this error localization, error can be detected during the storage correctness verification which acts as a data corruption. However, such important area reminds to be fully explored in the literature.

Cloud Computing contributions can be summarized into three main aspects –

- In comparison with numerous predecessors, which support binary results regarding the storage status across distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization.
- Beyond, considering the major and the minor prior works for ensuring remote data integrity the new scheme, effectively supports efficient security and active operations on data blocks.
- The try out results demonstrating the proposed scheme is highly applicable. Extended security analysis proves our scheme is resilient against tangled failure, malicious data modification attack and even server colluding attacks.

II. TROUBLESHOOTING STATEMENT

A. *Architecture Prototyping*



This model involves illustration of user essentiality and cloud service inhibitor.

USER: User may include authorized individual or an organization who have data to be stored in the cloud and rely on the cloud for data computational operations.

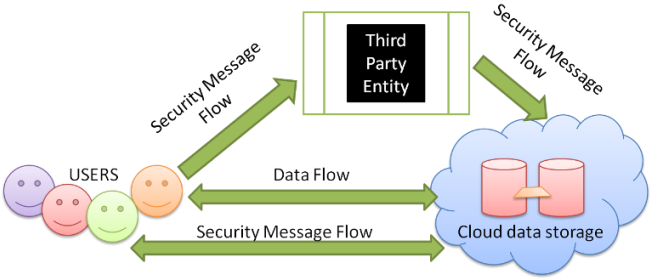
CLOUD SERVICE INHIBITOR: It has significant resources and expertise in building, maintenance and managing distributed cloud storage servers, owners and operates live cloud computing.

THIRD PARTY ENTITY: An optional entity who has expertise in capabilities that existing users may not have, is trusted to access and expose risk of cloud storage services on behalf of the users, upon predefined request.

A Simultaneous, co-operated and distributed manner of saved data is running through the cloud server inhibitor into a set of cloud servers where cloud data storage is maintained. Thereby data redundancy can be employed with a technique of erasure-correcting code to further tolerate mismatches or server crashes or existence of default as user's data can be uplifted in size or importance based criteria. Thereafter for application purposes, the user interacts with the cloud server through Cloud Server Inhibiter to access or to retrieve his own stored data.

As users no longer possess their data locally, it is of critical importance to assure them that their data are being correctly stored and maintained with the operations such as – insert, delete, update and affix. Now, user should be prepared with security means so that they can make continuous correctness assurance of their data stored in cloud servers even without the existence of local copies. In case those users do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the task to an optional trusted third part entity of their respective chosen option. Here, we evolve it with enterprise view fulfilling the task Peer-to-Peer communication channels between each cloud server and the user is reliable.

B. *Antagonist Model*



Threats faced by cloud data storage to ensure security come from two different sources one of them is cloud service inhibiter can be self-interested and un trusted and possibly malicious which is termed as external attacking. Then the other one is internal attacks. These internal attacks are hidden

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

and involved critical tasking and resolving them. Not only does it desire to move data that has not been or is rarely accessed to lower tier of storage than agreed for monitory reasons but it may also attempt to hide a data loss incident due to management errors, tangled failures. For external attacks data integrity threats may come from outsiders like – economically motivated attackers. They may compromise a numerous amounts of cloud data storage servers in different intervals of time and subsequently be able
to modify or delete users' data while remaining undetected by the inhibiter.

As we know this model refers to the act opposing someone. In accordance with it, it follows listed capabilities which capture both external and internal threats towards the make sure of data integrity maintained in the cloud. As well said according to its view it can be described as continuously corrupting the user's data files by introducing its own fraudulent data to prevent the original data being retrieved by the user. In fact, this is equivalent to internal attack service where all servers are assumed colluding together from the earlier stages of applications.

Specifically we consider two types of antagonist with different levels of capability in this paper such as –

- Weak Opponent
- Strong Opponent

Weak Opponent: Once a server is compromised, an opponent can pollute the original data files by performing illegal data base operations to prevent the original data from being retrieved by the user.

Strong Opponent: This is worst case of scenario headed.
Here we assume that the opponent can compromise all the storage servers so that he can intensionally modify the data files as long as they are internally consistent.

C. Excogitative Destinations

For any particular application to work efficiently we need to prepare a design through the whole process of its implementation can be made applicable and satisfying further usage. This designing process can also be even applied to data storage in cloud under the aforementioned opponent existence. To make sure the availability of providing security and dependability, this designing can be done in two formats –

- Static view of data verification
- Dynamic view of data verification

This paper deals with aiming of design efficient mechanisms by choosing dynamic view for dynamic data verification and database operations to achieve the required goals as follows-

Entropy Accuracy: To ensure users that their data is indeed stored appropriately and kept actively all the time in the cloud.

Quick Fixing of Errors in Data: It induces ineffective location of mal-functioning server when data loss or collision or mismatch or corruption has been detected.

Active Data Accompaniment: As we know that cloud data enables all sorts of localities to access the data. It may also include accessing the data in the remote areas in order to give support to all commodities, data is stored in the form of distributed fashion even to withstand dynamic fashion on application to be viewed as an active data. This means even if there is any modifications taken place in the database such as insertion, deletion, update, affix and append. On performing these operations also the data can be maintained in the active state for the further access in the cloud.

Reliability: To enhance data availability against tangled failures, malicious data modifications and server colluding attacks, minimising the effects brought by the data errors.

These design goals enable users to perform storage correctness checks with minimum load.

D. Preliminaries

- Entropy is taken in the form of string of data comprising all the characters of the user's need.
- Parameters are considered along with the key to provide security is operated on it to compute.
- Append is the main operation used to build the parameters considered.
- This helps in analysing the bit size, thereby to reduce the storage and make efficient if excess data is used.
- Fatal errors generated, thereby the failure of server if in case exist can overcome by allotted runtime exceptions provided.
- A secret key is used to encode the data into the required format so that the particular token is allotted to the existing data without any loss or affect causing to the entropy as it is recalled into its original format. So that it can be easily understood by the developer to enhance it much more better and recover it further if any problems exist.

III. SECURE DATA STORAGE IN CLOUD

In Cloud data storage system, users store their data in the cloud and no longer possess the data locally. Thus the correctness and availability of the stored data files on distributed cloud servers must be guaranteed. One of the keys issue is to effectively detect any unauthorized modification, collision and corruption due to server compromise or random tangled failures. This inconsistency can be successfully detected, to find with server the data error lies in of its great significance, since it can be the first step fast recover the storage error. In order to ensure cloud data storage, then a homomorphic or similar token is introduced. The token computation function, we are considering belongs to a family

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

of universal hash functions, chosen to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasure-coded data.

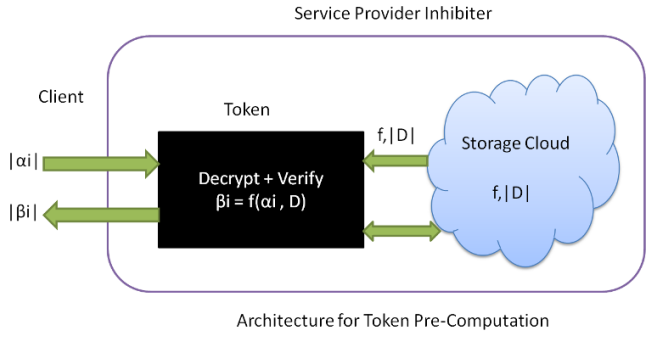
Subsequently, it is shown how to derive a challenge response protocol for verifying the storage correctness as well as identifying misbehaving servers. Finally, the procedure

forefile retrieval and error recovery based on erasure-correcting code is figured.

A. Finiteness and Fitness of Token Pre-Computation

To achieve assurance of data storage correctness and data error localization, our scheme entirely relies on the pre -computed verification tokens. The main idea is before file distribution the user pre -computes a certain number of short verification tokens on individual; each token covers a random subset of data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud,
Challenges the cloud servers with a set of randomly generated block indices. After getting assurance of the user it again asks for authentication by which the user is confirmed to be the authenticated user. Upon receiving assurance, each cloud server computes a short "signature" over the specified blocks and returns them to the user. The values of these signatures should match the corresponding tokens pre -computed by the user. Meanwhile, as all servers operate over the same subset of the indices, the requested response values for integrity check must also be a valid codeword determined by a secret matrix. Suppose the user wants to challenge the cloud server's t times to make sure the correctness of data storage. Then, he must pre -compute t verification tokens for each function, a challenge key and a master key are used. To generate the i^{th} token for server j, the user acts as follows –

- Derive a arbitrary value i and a permutation key based on master permutation key.
- Compute the set of randomly-chosen indices:
- Calculate the token using encoded file and the arbitrary value derived.



Architecture for Token Pre-Computation

Algorithm - Token Pre-computation

```
Algorithm computePreToken(Map params, key)
{
   // consider some existing elements to be parameters as to be entered, which are stored.
   // to TreeSet to be built consider strings which involve all characters
      names := params.keySet();
  // initialize StringBuilder for building the string by variable as string

  // using append command to buid the string by getting its size in bytes


 // this append operation is to check for all the data contained in the storage area of cloud for allotting tokens on further movement ahead
    For (name) then
   {
    if string_length > 0 then
    string_append;
    string_append with parameters of name;
    }
   return getHmac(string.toString(), key.getBytes());

//  Print pre- Token value using computePreAuth(params, key);
}

Algorithm String getHmac( data, key) {
// to do the hash function mappings to the data with the token allotment for reducing the redundant data formation.
          try {
             //initialize bytekey value
           bk: = key;
              }
   // create instance for considered Hash function Mac.getInstance("data");
              mac.init(bytekey);
              return toHex(mac.doFinal(data.getBytes()));
   catch exception as NoSuchAlgorithmException then
    {
              throw as
              RuntimeException - "fatal error";
    }
   catch exception as InvalidKeyException then
    {
              throw as
              RuntimeException - "fatal error";
    }
   }

Algorithm ByteKey implements SecretKey
  {
```

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

```
 //initialize mKey;
mKey := key.clone();
// on encoding completion
  return mKey;
 }
 // on performing hash function
  return "data";
// To get the original format even after conversion into encoded data
      return "RAW";   }}


Algorithm toHex(data){

        // update the length of the string which is already considered by doubling it with its original length
        String = data.length *2;
         for   i:=0 and i<data.length then
        {
           i=i+1;
           string.append(hex[(data[i] & 0xf0) >>> 4]);
           string.append(hex[data[i] & 0x0f] );
         }
          return string.toString(); }
```

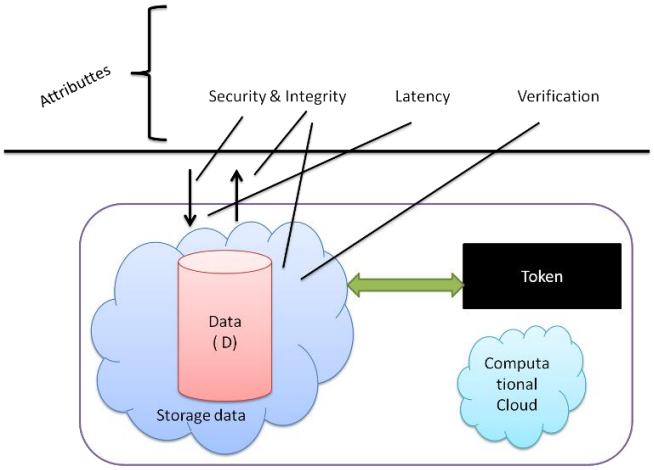
B. *Encryption and Authentication*

Providing security that is ensuring confidentiality, authenticity of data can be guaranteed either symmetrically on using single key or asymmetrically on using two independent keys.

The symmetric key can be instantiated with a combination of symmetric encryption and message authentication code which is given as getHmac( ). These themes use a respective symmetric key for encryption and same key for decryption. This could be used for example to construct an outsourced database to which new entries can be appended by multiple parties or users without using shared symmetric key which can be associated with public key cryptography where the usage of two separate keys for encryption and decryption exist.

C. *Fully Homomorphic Encryption*

Fully homomorphic encryption is semantically secure public-key encryption that additionally allows computing an arbitrary function on encrypted data using the public-key only, i.e., given a cipher text $|\alpha|$, a function f and the public-key pk, it is possible to compute $|\beta|$. Such that evaluating $|\beta|=EVAL_{pk}(f,|\alpha|)=|f(\alpha)|$. Constructing a homomorphic encryption scheme with polynomial overhead was a longstanding open problem. Here, in this notation $|\alpha|$ stands as homomorphically encrypted data α.



D. *Correctness Verification and Error Localization*

Error localization is a key prerequisite for eliminating errors in storage systems. However, many previous schemes do not explicitly consider the problem of data error localization, thus only provide binary results for the storage verification. Our scheme outperforms those by integrating the correctness verification and error localization in our challenge-response protocol: the response values from servers for each challenge not only determine the correctness of the distributed storage, but also contain information to locate potential data error(s).

Specifically, the procedure of the i^{th} challenge -response for a cross-check over the n servers is described as follows:

- The user reveals the i as well as the i^{th}
- key k (i) to each servers
- The server storing vector G aggregates those r rows
- Specified by index k(i) into a linear combination R
- Upon receiving R is from all the servers, the user takes away values in R.
- Then the user verifies whether the received values remain a valid codeword determined by secret matrix.

Here, R is a relation and G is a vector.

Algorithm: Correctness Verification and Error Localization

```
Algorithm  Verification(i)
{
     Recompute αi and Key
     Send {α,Key} to all the cloud servers
     Receive from servers R
     {
```

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

Relations which summarize the cipher text with the considered vector involving data partitions till its higher limit

```
If ((Ri(1).....Ri(m)).P==(Ri(m+1),....,Ri(n))) then
{
Accept and ready for the next challenge.
Else
Return server j is misbehaving.
}
}
}
```

IV. Providing Dynamic Data Operation Support

So far, we assumed that F represents archived data. However, in cloud data storage, there are many potential scenarios where data stored in the cloud is dynamic, like electronic documents, photos, or log files etc. Therefore, it is crucial to consider the dynamic case, where a user may wish to perform various block -level operations of revise, erase and affix to modify the data file while maintaining the storage correctness assurance.

The straightforward and insignificant way to support these operations is for user to download all the data from the cloud servers and re -compute the whole parity blocks as well as verification tokens. This would clearly be highly inefficient. In this section, we will show how our scheme can unambiguously and efficiently handle dynamic data operations for cloud data storage.

A. *Revise Operation*

In cloud data storage, sometimes the user may need to modify some data block(s) stored in the cloud, from its current value f to a new one. We refer to this operation as data revise.

B. *Erase Operation*

Sometimes, after being stored in the cloud, certain data blocks may need to be erased. The erase operation we are considering is a general one, in which user replaces the data block with zero or some special reserved data symbol. From this point of view, the erase operation is actually a special case of the data revise operation, where the original data blocks can be replaced with zeros or some predetermined special blocks.

C. *Append Operation*

In some cases, the user may want to increase the size of his stored data by adding blocks at the end of the data file , which we refer as data append. We anticipate that the most frequent append operation in cloud data storage is bulk append, in which the user needs to upload a large number of blocks (not a single block) at one time.

D. *Affix Operation*

An affix operation to the data file refers to an affix operation at the desired index position while maintaining the same data block structure for the whole data file, i.e., inserting a block F corresponds to shifting all blocks starting with index j + 1 by one slot. An affix operation may affect many rows in the logical data file matrix F , and a substantial number of computations are required to renumber all the subsequent blocks as well as re -compute the challenge-response tokens. Therefore, an efficient affix operation is difficult to support and thus we leave it for our future work.

V. Security Analysis and Performance Issues

A. *Security Strength against weak opponent*

i) Detection Probability against data modification:

In our scheme, servers are required to operate on specified list of tokens. These selected tokens greatly reduce the computational overhead on the server, while maintaining the detection of the data corruption with high probability. Note that if none of the specified r rows in the i^{th} verification process are erased or modified, the antagonist avoids the detection.

ii) Identification Probability for Misbehaving Servers:

We have shown that, if the antagonist modifies the data blocks among any of the data storage servers, our sample checking scheme can successfully detect the attack with high probability. As long as the data modification is caught, the user will further determine which server is malfunctioning. This can be achieved by comparing the response values R with the pre-stored tokens v. The probability for error localization or identifying misbehaving server(s) can be computed in a similar way. It is the product of the matching probability for sampling check and the probability of complementary event for the false negative result. Next, we consider the fake denial probability that R(j)=v(j) when at least one of z blocks are modified. Thus, the identification probability for misbehaving server(s) is predicted.

B. *Security Strength against Strong opponent*

We analyze the security strength of our schemes against server colluding attack and explain why blinding the parity blocks can help improve the security strength of our proposed scheme. Redundancy parity vectors are calculated via multiplying the file matrix F by P , where P is the secret parity generation matrix we later rely on for storage correctness assurance. If we disperse all the generated vectors directly after token pre -computation, i.e., without blinding, malicious servers that collaborate can reconstruct the secret P matrix easily: they can pick blocks from the same rows among the data and parity vectors to establish a set of m * k linear equations and solve for the m · k entries of the parity generation matrix P . Once they have the knowledge of P, those malicious servers can consequently modify any part of the data blocks and calculate the corresponding parity blocks, and vice versa, making their codeword relationship always

consistent. Therefore, our storage correctness challenge scheme would be damaged even if those modified blocks are covered by the specified rows, the storage correctness check equation would always hold. To prevent colluding servers from recovering P and making up consistently -related data and parity blocks, we utilize the technique of adding random perturbations to the

encoded file matrix and hence hide the secret matrix P . We make use of a keyed pseudorandom function f with key k, both of which has been introduced previously.

C. *Performance evaluation*

File Distribution Preparation is implemented for the generation of parity vectors for our scheme. This experiment is conducted using JAVA on a system with an Intel Core 2 processor running at 1.86 GHz, 2048 MB of RAM and 250 GB Serial ATA drive. Thus the cost decreases when more

data vectors are involved. The performance of our scheme is comparable and even our scheme supports dynamic data operation while is for static data only. Verification Token Pre - computation: In our scheme we use fixed number of verification token t that are determined before file distribution, we can overcome this issue by choosing sufficient large token in practice.

VI. CONCLUSION

In this paper, we studied the problem of data security in data storage in cloud servers. To guarantee the correctness of users' data in cloud data storage, we proposed an effectual and flexible scheme with explicit dynamic data support, including block revise, erase, and affix. We use erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. Our scheme accomplishes the integration of storage correctness insurance and data corruption has been detected during the storage correctness verification across the distributed servers. Our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks. We believe that data storage security in Cloud Computing, an area full of challenges and of dominant significance, is still in its infancy t o be identified. We envision several possible directions for future research on this area. It allows Third Parity Auditor to audit the cloud data storage without demanding users' time, probability.

REFERENCES

[1] [1] Robert Gellman and World Privacy Forum , "Privacy in the Clouds: Risks to Privacy and Confidentiality from Cloud Computing", February 23, 2009.
[2] Amazon.com, "Amazon Web Services (AWS)," Online at http://aws. amazon.com, Version 2010.
[3] Weiss, Aaron. Computing in the clouds. netWorker 11, 4 (Dec. 2007),
[4] <http://doi.acm.org/10.1145/1327512.1 327513>.
[5] [4] Eric A. Marks, Bob Lozano "Executive's Guide to Cloud computing", John Wiley & Sons, Inc.
[6] [5] Theart of Service, "A Complete Guide to Cloud Computing", http://theartofservice.com.
[7] [6] Tim Mather, Subra Kumaraswamy, and Shahed Latif, "Cloud Security and Privacy", Published by
[8] O'Reilly Media, Inc.,- 2009 .
[9] [7] Brian J.S. Chee and Curtis Franklin, Jr., "Cloud Computing, Technologies and Strategies of the
[10] Ubiquitous Data Center", CRC Press 2010 by Taylor and Francis Group, LLC.
[11] [8] N.Gohring, "Amazon's S3 down for several hours," Online at
[12] http://www.pc wor ld .co m/ bus iness cent er/ a rti cle /142549/amazons s3 down for several hours.html, 2008 .
[13] [9] Juels and J. Burton S. Kaliski, "PORs: Proofs of Retrievability for Large Files," Proc. of CCS '07, 2007.
[15] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. of SecureComm '2008.
[16] John W. Rittinghouse, James F. Ransome, " Cloud Computing Implementation, Management, and Security", CRC Press 2010 by Taylor and Francis Group, LLC.
[17] Journal of Theoretical and Applied Information Technology, "CLOUD COMPUTING".

Mr.A.V.Praveen Krishna currently working as Assistant Professor in department of CSE, K L University and pursuing the Ph.D in Computer Science & Engineering. He is actively engaged in research and publications in the areas of Network Security, Data Mining, Green Computing & Cloud Computing. He completed his M.Tech (CSE) from K.L College of Engineering, Acharaya Nagarjuna University and secured First Rank. He is topper of the university and honoured with gold medal for his extraordinary performance in MCA from SCSVMV University, Tamilnadu. He was awarded "Significant Contribution Award" by CSI in 2011.In KL University he has feedback as an excellent teacher. Mr.Krishna is the Life member of CSI,IEEE & ISCA. Presently,he is Student Branch Counselor of CSI-K L University Student Branch.

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 6, August 2012

Ms. Bhavya Boggavarapu currently studying in the Department Of Computer Science and Engineering, K L University. She currently secured the top rank and stud as the topper of the Department of CSE with a grading point of 9.7 for 10 in the current semester. She has received two Gold Medals for the position of AIR-162 in National Science Olympiad (NSO) in 2007 and in National Interactive Maths Olympiad (NIMO). She has also got selected in AMTI – Association of Maths Teachers of India. She has published paper in National Conference. She has done a Summer Internship in the name of Summer Student Program 2012 held at Indian Institute of Science, Education and Research (IISER) Pune,Maharastra in the area of Mathematics which can be applied in Cryptography, topic of Computer Networks.

Ms. Baby Sindhura Katragadda studying in the Department Of Electronics and Communication Engineering, K L University. She currently secured the top position and stud as one of the toppers of the department of ECE with a grading point of 9.3 for 10 in the current semester. She has published a paper in National Conference – 2012.