ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

Efficient Remote Data Possession Checking In Critical Information Infrastructures Ensuring Data Storage Security In Cloud Computing

Dr. T.Nalini [1] , Dr.K.Manivannan[2],Vaishnavi Moorthy[3]

[1] Professor, Department of Computer Science& Engineering, Bharath University, Chennai, TN, India

[2] Professor, Department of MCA, RMK Engineering College, Chennai, TN, India

[3]Assistant Professor, Dept. of CSE, Bharath University, TN, India

ABSTRACT: Cloud computing has been envisioned as the on-demand self-service, ubiquitous network access, location independent resource pooling, rapid resource elasticity, usage-based pricing and transference of risk. Today, technical research works focus on Remote data possession Checking protocols permit to check that a remote server can access an uncorrupted file with the help of third party verifiers. In this paper, Seb´e et al.'s protocol is adapted to support efficient remote data possession checking in critical information infrastructure without the help of a third party auditor. This design allows users to audit the cloud storage with very lightweight communication and computation cost. In addition, the auditing result not only ensures strong cloud storage correctness guarantee, but also simultaneously achieves fast data error localization, i.e., the identification of misbehaving remote server. The design further supports secure and efficient dynamic operations on outsourced data, including block modification, deletion, and append. Through a formal analysis, the correctness and security of the protocol is shown. The proposed scheme is highly efficient and resilient against the malicious data modification attack, server clouding attacks and failure.

KEYWORDS*:* Cloud Computing, Remote data possession Checking, third party auditor, Seb´e et al.'s protocol, server clouding attacks

I. INTRODUCTION

Cloud computing is the long dreamed vision of computing as a utility, where users can remotely store their data into the cloud so as to enjoy high quality applications and services from a shared pool of configurable computing resources (fig 1). It works on a client-server basis, using web browser protocols.

A cloud user needs a client device such as a laptop or desktop computer, pad computer, smart phone, or other computing resource with a web browser (or other approved access route) to access a cloud system via the World Wide Web. Typically the user will log into the cloud at a service provider or private company, such as their employer. The cloud provides server-based applications and all data services to the user, with output displayed on the client device. Memory allocated to the client system's web browser is used to make the application data appear on the client system display, but all computations and changes are recorded by the server, and final results including files created or altered are permanently stored on the cloud servers.

Performance of the cloud application is dependent upon the network access, speed and reliability as well as the processing speed of the client device [2]. While Cloud Computing makes these advantages more appealing than ever, it also brings new and challenging security threats towards users' outsourced data. Since cloud service providers (CSP) are separate administrative entities, data outsourcing is actually relinquishing user's ultimate control over the fate of their data.

As a result, the correctness of the data in the cloud is being put at risk due to the following reasons. First of all, although the infrastructures under the cloud are much more powerful and reliable than personal computing devices, they are still facing the broad range of both internal and external threats for data integrity [3][4][5]. Secondly, for the benefits of their own, there do exist various motivations for cloud service providers to behave unfaithfully towards the cloud users regarding the status of their outsourced data [6][7]. These problems, impedes the successful deployment of the cloud architecture.

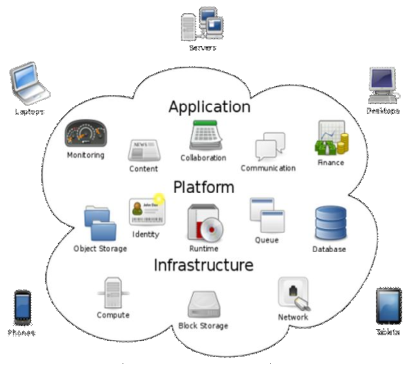

Fig:1 Cloud Computing

The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centres. Hence, a lot of works [16] have been done on designing remote data possession checking protocols, which allow data integrity to be checked without completely downloading the data. These protocols support data dynamics at the block level, including block insertion, block modification and block deletion, support public verifiability, by which anyone (not just the client) can perform the integrity checking operation against third party verifiers. In addition the protocol should achieve the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, the scheme should almost guarantee the identification of the misbehaving server(s) for effective cloud storage.

In this paper, the main contributions of the proposed Seb´e et al.'s protocol are:

A remote data possession checking protocol for cloud storage can be viewed as an adaptation of Seb´e et al.'s protocol [14]. The proposed protocol inherits the support of remote data possession checking and supports public verifiability and privacy against third party verifiers, while at the same time it doesn't need to use a third-party auditor.A security analysis of the proposed protocol, which shows that it is secure against the untrusted server and private against third party verifiers, is given [14].Compared to many of its predecessors, which only provide binary results about the storage status across the distributed servers, the proposed scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s) [15].

Unlike most prior works for ensuring remote data integrity, the new scheme further supports secure and efficient dynamic operations on data blocks, including: update, delete and append [14]. The research paper is organised as Section II describing the existing system and proposed system. Section III outlines the proposed protocol functions and Section IV lists the related work and Section V concludes the paper with future work.

II. BACKGROUND WORK

In the existing system cloud data storage service involve three different entities, the cloud user (U), who has large amount of data files to be stored in the cloud; the cloud server (CS), which is managed by cloud service provider (CSP) to provide data storage service; the third party auditor (TPA), who has expertise and capabilities that cloud users do not have and is trusted to assess the cloud storage service security on behalf of the user upon request. Users rely on the CS for cloud data storage and maintenance (fig 2). The TPA, who is in the business of auditing, is assumed to be reliable and independent, and thus has no incentive to collude with either the CS or the users during the auditing process. TPA should be able to efficiently audit the cloud data storage without local copy of data and without bringing in additional on-line burden to cloud users. However, any possible leakage of user's outsourced data towards TPA through the auditing protocol should be prohibited. To achieve the audit delegation and authorize CS to respond to TPA's audits, the user can sign a certificate granting audit rights to the TPA's public key, and all audits from the TPA are authenticated against such a certificate. For external attacks, data integrity threats may come from outsiders who are beyond the control domain of CSP, for example, the economically motivated attackers. They may compromise a number of cloud data storage servers in different time intervals and subsequently be able to modify or delete users' data while remaining undetected by CSP.

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

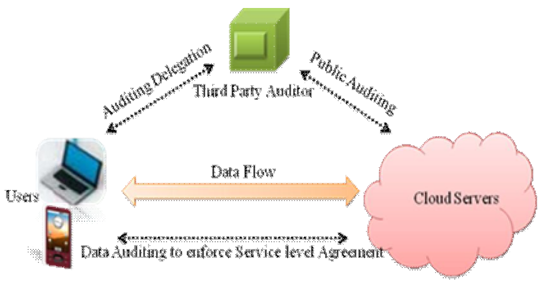

Fig.2 Existing System

In proposed system an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. The adaptation of proposed protocol- Seb´e et al.’s protocol with distributed verification of erasure-coded data, the scheme achieves the public verifiability and data dynamics against the third party verifiers which shows the detection of data corruption during the storage correctness verification across the distributed servers. The protocol design will achieve the following security and performance guarantee: 1) Public auditability 2) Storage correctness 3) Privacy-preserving 4) Batch auditing 5) Lightweight. The model we propose aims to protect cloud data against untrusted service providers. This model involves data owners, cloud service providers, and data users. Data owners store data in the cloud and send every share of data entries to the service providers. Data users access data from the service providers and have access to the Public information of data owners in order to verify the shares received from service providers. Fast localization of data error is to effectively locate the malfunctioning server when data corruption has been detected.

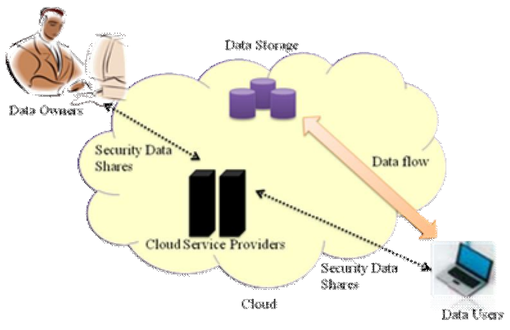

Fig.3 Proposed System

III. PROTOCOL ANALYSIS

A cloud storage system in which there is a client and an untrusted server is considered. The client stores her data in the server without keeping a local copy. Hence, it is of critical importance that the client should be able to verify the integrity of the data stored in the remote untrusted server. If the server modifies any part of the client’s data, the client should be able to detect it and should not be detected by any third party verifier. In this case, when a third party verifier verifies the integrity of the client’s data, the data should be kept private against the third party verifier.

The proposed protocol is correct in the sense that the server can pass the verification of data integrity as long as both the client and the server are honest. Then the protocol is secure against the untrusted server. The protocol guarantee is that, assuming the client is honest, if and only if the server has access to the complete and uncorrupted data, it can pass the verification process successfully. Finally the protocol is private against third party verifiers. To design the remote data integrity checking, Seb´e et al.’s protocol the following five functions needed are

(a) SetUp,
(b) TagGen,
(c) Challenge
(d) Gen-Proof
(e) Check-Proof

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

Let m be the file that will be stored in the untrusted server, which is divided into n blocks of equal lengths: $m = m_1, m_2 \dots m_n$, where $n = \lceil |m|/l \rceil$. Here l is the length of each file block. Denote by $f_K(\cdot)$ a pseudo-random function which is defined as: f : {0, 1}k × {0, 1}log2(n) → {0, 1}d, in which k and d are two security parameters. Furthermore, denote the length of N in bits by |N|.

(a) ***SetUp*** $(1^k) \rightarrow$ (pk, sk): Given the security parameter k, this function generates the public key pk and the secret key sk. pk is public to everyone, while sk is kept secret by the client.

(b) ***TagGen*** (pk, sk,m) → Dm: Given pk, sk and m, this function computes a verification tag D_m and makes it publicly known to everyone. This tag will be used for public verification of data integrity.

(c) ***Challenge*** (pk, D_m) → chal: Using this function, the verifier generates a challenge chal to request for the integrity proof of file m. The verifier sends chal to the server.

(d) ***GenProof*** $(pk, D_m, m, chal)$ → R: Using this function, the server computes a response R to the challenge chal. The server sends R back to the verifier.

(e) ***CheckProof*** $(pk, D_m, chal, R)$ → {“success”, “failure”}: The verifier checks the validity of the response R. If it is valid, the function outputs “success”, otherwise the function outputs “failure”. The secret key sk is not needed in the CheckProof function. These functions are used for data dynamics.

(f) ***CheckMisbehave***(r, en_f, m') → n: Let r be the number of different rows for which the user asks for checking in a challenge for the encrypted file matrix en_f and m’ be the matching factor. Using the function, the verifier can detect the unusual behaving server and if none of the specified rows in the process are deleted or modified, the adversary avoids the detection.

There are three security requirements for the remote data possession checking protocol and an algorithm for the file retrieval and error recovery from the affected servers:

1. Security against the server with public verifiability
2. Privacy against third party verifiers.
3. Identifying misbehaviour server
4. File retrieval and error recovery

When the verifier is not the client herself, the protocol must ensure that no private information about the client’s data is leaked to the third party verifier.

1) *Security against the Server with Public Verifiability*

If $CheckProof(pk, D_m, chal, R)$ = “success”, then the remote data possession checking protocol is said to be secure against the server for any PPT (probabilistic polynomial time).

2) *Privacy against Third Party Verifiers*

For the remote data possession checking protocol Π, if there exists a PPT simulator S_V such that $\{S_V(x, f_V(x, y))\}_{x,y} € \{0,1\}^* \equiv \{view\ \Pi_V(x, y)\}_{x,y} € \{0,1\}^*$, then Π is the protocol that ensures privacy against third-party verifiers, where ≡ denotes computational indistinguishability.

3) *Identification Probability for Misbehaving Servers*

From user’s perspective, the model has to capture all kinds of threats towards the cloud data integrity. Because cloud data do not reside at user’s local site but at CSP’s address domain, these threats can come from two different sources: internal and external attacks. For internal attacks, the domain can be self-interested, untrusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors. For external attacks, data integrity threats may come from outsiders who are beyond the control domain, for example, the economically motivated attackers. They may compromise a number of cloud data storage servers in different time

ISSN (Print) : 2320 – 9798
ISSN (Online): 2320 – 9801

International Journal of Innovative Research in Computer and Communication Engineering
Vol. 1, Issue 1, March 2013

intervals and subsequently be able to modify or delete users' data while remaining undetected. Therefore, the proposed model has the capabilities, which captures both external and internal threats towards the cloud data integrity.

Suppose n servers are misbehaving due to the possible compromise failure, assume the adversary modifies the data blocks in z rows out of the l rows in the encoded file matrix. Let r be the number of different rows for which the user asks for checking in a challenge. Let X be a discrete random variable that is defined to be the number of rows chosen by the user that matches the rows modified by the adversary. The matching probability that at least one of the rows picked by the user matches one of the rows modified by the adversary is analysed first.

$$\mathbf{P^{r}_{m'} = 1\text{-}P\{X=0\} = 1-\prod_{i=0}^{r-1}\left(1 - \min\left\{\frac{z^{\wedge}}{l-i},1\right\}\right) \geq 1\text{-}\left(\frac{l^{m}-z}{l}\right)^{r}}$$

If none of the specified r rows in the i^{th} verification process are deleted or modified, the adversary avoids the detection. This can be achieved by comparing the response values $R_i^{(j)}$ with the pre-stored tokens $v_i^{(j)}$, where j € {1, . . . , n}. The probability for error localization or identifying misbehaving server(s) is 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.

$$\mathbf{{}^{\wedge}P^{r}_{m'} = 1-\prod_{i=0}^{r-1}\left(1 - \min\left\{\frac{z^{\wedge}}{l-i},1\right\}\right)}$$

The matching probability is where $z^\wedge \leq z$, $^\wedge\ P^{r}_{m'}$ matching probability of the modified rows in encoded file matrix. Next, the false negative probability P^{r}_{f} is considered such that $R_i^{(j)} = v_i^{(j)}$ when at least one of z^ blocks is modified. When two different data vectors collide, the probability is

$$\mathbf{{}^{\wedge}P^{r}_{f} = 2^{-p}}$$

Thus, the identification probability for misbehaving server(s) is

$$\mathbf{{}^{\wedge}P_{d} = {}^{\wedge}P^{r}_{m'} \,.\, (1 - {}^{\wedge}P^{r}_{f})}$$

where P_d is the detection probability against data modification. The above formulation for localization of a misbeHaving server is integrated to the remote data integrity checking, Seb´e et al.'s protocol, thus making it more efficient and secured protocol. The protocol can be easily extended into a probabilistic one by using the probabilistic framework. The proposed protocol has very good efficiency in the aspects of communication, computation and storage costs [17].

4) *File Retrieval and Error Recovery*

Whenever the data corruption is detected, the comparison of pre-computed tokens and received response values can guarantee the identification of misbehaving server(s). Therefore, the user can always ask servers to send back blocks of the r rows specified in the challenge and regenerate the correct blocks by the erasure correction as long as the number of identified misbehaving servers is less than k. The newly recovered blocks can then be redistributed to the misbehaving servers to maintain the correctness of storage.

Algorithm:

```
1: procedure
   // assume the block corruptions have been detected among the specified r rows and
   // assume $s \leq k$ servers have been identified as misbehaving servers
2. Download r rows of blocks from servers;
3. Treat s servers as erasures and recover the blocks with k as a security parameter
4. Resend the recovered blocks to corresponding servers
5. end procedure
```

III. RELATED WORK

I. Shah et al. [8],[9] propose allowing a TPA to keep online storage honest by first encrypting the data then sending a number of pre-computed symmetric-keyed hashes over the encrypted data to the auditor. The auditor verifies both the integrity of the data file and the server's possession of a previously committed decryption key. This scheme only works for encrypted files and it suffers from the auditor statefulness and bounded usage, which may potentially bring in on-line burden to users when the keyed hashes are used up.

II. Ateniese et al. [6] were the first who defined the "provable data possession" (PDP) model for ensuring possession of file on untrusted storages. . Their scheme utilizes the RSA-based homomorphic authenticators for auditing outsourced data and suggests randomly sampling a few blocks of the file. However, the public auditability in their scheme demands the linear combination of sampled blocks exposed to external auditor. When used directly, their protocol is not provably privacy preserving, and thus may leak user data information to the auditor.

III. In their subsequent work, Ateniese et al. [10] described a PDP scheme that uses only symmetric key based cryptography. This method has lower-overhead than their previous scheme and allows for block updates, deletions and appends to the stored file, which has also been supported in our work. However, their scheme focuses on single server scenario and does not provide data availability guarantee against server failures, leaving both the distributed scenario and data error recovery issue unexplored. The explicit support of data dynamics has further been studied in the two recent works [11] and [12].

IV. Schwarz et al. [13] proposed to ensure static file integrity across multiple distributed servers, using erasure-coding and block-level file integrity checks. Some ideas of their distributed storage verification protocol are being adopted. However, the scheme further support data dynamics and explicitly studies the problem of misbehaving server identification, while theirs did not.

V. Zhuo Hao et.al [14] proposed the remote data integrity checking protocol that supports public verifiability without the support of TPA and compared the properties of the proposed protocol with the then existing protocols.

VI. Wang et al.[15] in their work proposed a flexible distributed cloud storage integrity auditing mechanism utilizing the homomorphic token and distributed erasure coded data that detects the Byzantine failure, malicious data modification attack and server clouding attacks.

All the above schemes provide efficient methods for secured data verifiability, data storage integrity and detection of server attacks in the cloud based storage separately with an idea proposed for file retrieval and error recovery. In this paper the proposed Seb'e et al's protocol combines the mentioned characteristic functions together making it more efficient and secured when compared to other protocols.

V. CONCLUSION & FUTURE WORK

In this paper, a privacy-preserving protocol for remote data storage in the cloud is been proposed investigating the remote data possession checking. The focus is on stopping data being disclosed by un-trusted service providers when data owners distribute their database entries along with error recovery. To achieve the assurances of cloud data integrity and availability and enforce the quality of dependable cloud storage service for users, an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete and append is being implemented. The future research aims to extend the protocol to support data level dynamics at minimal costs to foster a trusted data transaction innovation.

REFERENCES

1. P. Mell and T. Grance, "Draft nist working definition of cloud computing," Referenced on June. 3rd, 2009 Online at http://csrc.nist.gov/groups/SNS/cloud-computing/index.html, 2009
2. www.google/wikipedia/cloud_computing
3. Amazon.com, "Amazon s3 availability event: July 20, 2008," Online at http://status.aws.amazon.com/s3-20080720.html, July 2008.
4. S.Wilson,"Application engine outage,"Online- http://www.cio-weblog.com/50226711/appengine outage.php, June 2008.
5. B.Krebs, "Payment Processor Breach May Be Largest Ever," Online at http://voices.washingtonpost.com/securityfix/2009/01/payment processor breach may b.html, Jan. 2009.
6. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," Cryptology ePrint Archive, Report 2007/202, 2007, http://eprint.iacr.org/.
7. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. of ESORICS'09, Saint Malo, France, Sep. 2009.
8. M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," Cryptology ePrint Archive, Report 2008/186, 2008, http://eprint.iacr.org/.
9. M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in Proc. of HotOS'07. Berkeley,CA, USA: USENIX Association, 2007, pp. 1–6.
10. G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in Proc. of SecureComm'08, 2008, pp. 1–10.
11. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in Proc. of ESORICS'09, volume 5789 of LNCS. Springer- Verlag, Sep. 2009, pp. 355–370.
12. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in Proc. of CCS'09, 2009, pp. 213–222.
13. T. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in Proc. of ICDCS'06, 2006, pp. 12–12.

14. Zhuo Hao, Sheng Zhong, Member, IEEE, and Nenghai Yu, Member, ”A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability”, IEEE-2011
15. “Towards Secure and Dependable Storage Services in Cloud Computing” Cong Wang, Student Member, IEEE, Qian Wang, Student Member, IEEE, Kui Ren, Member, IEEE,Ning Cao, Student Member, IEEE, and Wenjing Lou, Senior Member, IEEE-2011
16. Wang, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for data storage security in cloud computing,” in InfoCom2010, IEEE, March 2010.
17. “Implementing Remote Data Integrity Checking Protocol for Secured Storage Services with Data Dynamics and Public Verifiability in Cloud Computing”- Ms. Vaishnavi Moorthy & Dr. S.Sivasubramaniam from International Organization of Scientific Research Journal of Engineering (IOSRJEN), Mar.2012, Vol.2(3) Pages:496-500

.