www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 10 October, 2013 Page No. 2936-2943

Empowering Auditability of Public and Data Dynamics for Depot Protection in Cloud

Prof.Santosh Kumar. B[1], Vijaykumar .L[2]

[1]Department of computer science and engineering, Poojya doddappa Appa College of engineering, Gulbarga, Karnataka, India, email: santosh.bandak@gmail.com

[2]Department of computer science and engineering, Poojya doddappa Appa College of engineering, Gulbarga, Karnataka, India, email:vijay86.lak@gmail.com

Abstract: ***Cloud Computing has been envisioned as the next-generation architecture of IT Enterprise. It moves the application software and databases to the centralized large data centers, where the management of the data and services may not be fully trustworthy. This unique paradigm brings about many new security challenges, which have not been well understood. This work studies the problem of ensuring the integrity of data storage in Cloud Computing. In particular, we consider the task of allowing a third party auditor (TPA), on behalf of the cloud client, to verify the integrity of the dynamic data stored in the cloud. The introduction of TPA eliminates the involvement of the client through the auditing of whether his data stored in the cloud are indeed intact, which can be important in achieving economies of scale for Cloud Computing. The support for data dynamics via the most general forms of data operation, such as block modification, insertion, and deletion, is also a significant step toward practicality, since services in Cloud Computing are not limited to archive or backup data only. While prior works on ensuring remote data integrity often lacks the support of either public Auditability or dynamic data operations, this paper achieves both. We first identify the difficulties and potential security problems of direct extensions with fully dynamic data updates from prior works and then show how to construct an elegant verification scheme for the seamless integration of these two salient features in our protocol design. In particular, to achieve efficient data dynamics, we improve the existing proof of storage models by manipulating the classic Merkle Hash Tree construction for block tag authentication. To support efficient handling of multiple auditing tasks, we further explore the technique of bilinear aggregate signature to extend our main result into a multi-user setting, where TPA can perform multiple auditing tasks simultaneously. Extensive security and performance analysis show that the proposed schemes are highly efficient and provably secure.***

Keywords: data dynamics, public auditability, data storage, cloud computing**.**

1. Introduction

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the "software as a service" (SaaS) computing architecture, are transforming data centers into pools of computing service on a huge scale. Meanwhile, the increasing network bandwidth and reliable yet flexible network connections make it even possible that clients can now subscribe high-quality services from data and software that reside solely on remote data centers. Although envisioned as a promising service platform for the Internet, this new data storage paradigm in "Cloud" brings about many challenging design issues which have profound influence on the security and performance of the overall system. One of the biggest concerns with cloud data storage is that of data integrity verification at untrusted servers. For example, the storage service provider, which experiences Byzantine failures occasionally, may decide to hide the data errors from the clients for the benefit of their own. What is more serious is that for saving money and storage space the service provider might neglect to keep or deliberately delete rarely accessed data files which belong to an ordinary client. Consider the large size of the outsourced electronic data and the client's constrained resource capability, the core of the problem can be generalized as how can the client find an efficient way to perform periodical integrity verifications without the local copy of data files.

In order to solve the problem of data integrity checking, many schemes are proposed under different systems and security models . In all these works, great efforts are made to design solutions that meet various requirements: high scheme efficiency, stateless verification, unbounded use of queries and retrievability of data, etc.

Considering the role of theverifier in the model, all the schemes presented before fall into two categories: private auditability and public auditability. Although schemes with private auditability can achieve higher scheme efficiency, public auditability allows anyone, not just the client (data owner), to challenge the cloud server for correctness of data storage while keeping no private information. Then, clients are able to

delegate the evaluation of the service performance to an independent third party auditor (TPA), without devotion of their computation resources. In the cloud, the clients themselves are unreliable or may not be able to afford the overhead of performing frequent integrity checks. Thus, for practical use, it seems more rational to equip the verification protocol with public auditability, which is expected to play a more important role in achieving economies of scale for Cloud Computing.

Moreover, for efficiency consideration, outsourced data themselves should not be required by the verifier for the verification purpose. Another major concern among previous designs is that of supporting dynamic data operation for cloud data storage applications. In Cloud Computing, the remotely stored electronic data might not only be accessed but also updated by the clients, e.g., through block modification, deletion, insertion, etc. Unfortunately, the state of the art in the context of remote data storage mainly focus on static data files and the importance of this dynamic data updates has received limited attention so far . Moreover, as will be shown later, the direct extension of the current provable data possession (PD schemes to support data dynamics may lead to security loopholes. Although there are many difficulties faced by researchers, it is well believed that supporting dynamic data operation can be of vital importance to the practical application of storage outsourcing services. In view of the key role of public auditability and data dynamics for cloud data storage, we propose an efficient construction for the seamless integration of these two components in the protocol design.

Our contribution can be summarized as follows:

1. We motivate the public auditing system of data storage security in Cloud Computing, and propose a protocol supporting for fully dynamic data operations, especially to support block insertion, which is missing in most existing schemes.
2. We extend our scheme to support scalable and efficient public auditing in Cloud Computing. In particular, our scheme achieves batch auditing where multiple delegated auditing tasks from different users can be performed simultaneously by the TPA.
3. We prove the security of our proposed construction and justify the performance of our scheme through concrete implementation and comparisons with the state of the art.

1.1 Related Work

Recently, much of growing interest has been pursued in the context of remotely stored data verification Ateniese was the first to consider public auditability in their defined "provable data possession" model for ensuring possession of files on untrusted storages. In their scheme, they utilize RSA-based homomorphic tags for auditing outsourced data, thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems. In their subsequent work , Ateniese et al. propose a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported. Wang et al. consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors. Similar to [12], they nly consider partial support for dynamic data operation. Juels and Kaliski [3] describe a "proof of retrievability" model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on archive service systems. Specifically, some special blocks called "sentinels" are randomly embedded into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [12], the number of queries a client can perform is also a fixed priori, and the introduction of precomputed "sentinels" prevents the development of realizing dynamic data updates. In addition, public auditability is not supported in their scheme. Shacham and Waters [4] design an improved PoR scheme with full proofs of security in the security model defined in [3]. They use publicly verifiable homomorphic authenticators built from

BLS signatures [16], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static data files. Erway et al. [14] were the first to explore constructions for dynamic provable data possession.

They extend the PDP model in [2] to support provable updates to stored data files using rank-based authenticated skip lists. This scheme is essentially a fully dynamic version of the PDP solution. To support updates, especially for block insertion, they eliminate the index information in the "tag" computation in Ateniese's PDP model [2] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear. Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing.

Portions of the work presented in this paper have previously appeared as an extended abstract [1]. We revise the paper a lot and add more technical details as compared to [1]. First, in Section 3.3, before the introduction of our proposed construction, we present two basic solutions (i.e., the MAC-based and signature-based schemes) for realizing data auditability and discuss their demerits in supporting public auditability and data dynamics. Second, wegeneralize the support of data dynamics to both PoR and PDP models and discuss the impact of dynamic data operations on the overall system efficiency both. In particular, we emphasize that while dynamic data updates can be performed efficiently in PDP models more efficient protocols need to be designed for the update of the encoded files in PoR models.

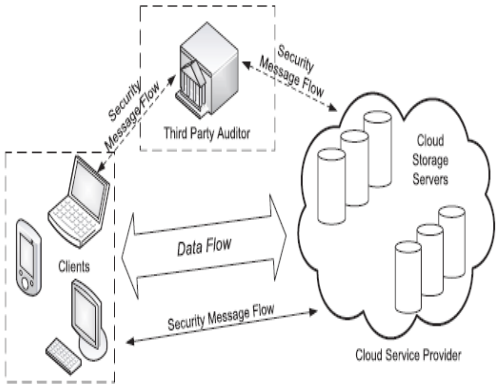


Fig 1: cloud data storage architecture

2. Reserch Elaborations

Ateniese et al. [2] are the first to consider public auditability in their defined "provable data possession" model for ensuring possession of files on untrusted storages. In their scheme, they utilize RSA-based homomorphic tags for auditing outsourced data, thus public auditability is achieved. However, Ateniese et al. do not consider the case of dynamic data storage, and the direct extension of their scheme from static data storage to dynamic case may suffer design and security problems.

In their subsequent work [12], Ateniese et al. propose a dynamic version of the prior PDP scheme. However, the system imposes a priori bound on the number of queries and does not support fully dynamic data operations, i.e., it only allows very basic block operations with limited functionality, and block insertions cannot be supported.

In [13], Wang et al. consider dynamic data storage in a distributed scenario, and the proposed challenge-response protocol can both determine the data correctness and locate possible errors.

Similar to [12], they only consider partial support for dynamic data operation. Juels and Kaliski [3] describe a "proof of retrievability"model, where spot-checking and error-correcting codes are used to ensure both "possession" and "retrievability" of data files on archive service systems. Specifically, some special blocks called "sentinels" are randomly embedded into the data file F for detection purpose, and F is further encrypted to protect the positions of these special blocks. However, like [12], the number of queries a client can perform is also a fixed priori, and the introduction of precomputed "sentinels" prevents the development of realizing dynamic data updates. In addition, public Auditability is not supported in their scheme.

Shacham and Waters [4] design an improved PoR scheme with full proofs of security in the security model defined in [3]. They use publicly verifiable homomorphic authenticators built from BLS signatures [16], based on which the proofs can be aggregated into a small authenticator value, and public retrievability is achieved. Still, the authors only consider static data files. Erway et al. [14] were the first to explore constructions for dynamic provable data possession. They extend the PDP model in [2] to support provable updates to stored data files using rank-based authenticated skip lists. This scheme is essentially a fully dynamic version of the PDP solution.

To support updates, especially for block insertion, they eliminate the index information in the "tag" computation in Ateniese's PDP model [2] and employ authenticated skip list data structure to authenticate the tag information of challenged or updated blocks first before the verification procedure. However, the efficiency of their scheme remains unclear.

Although the existing schemes aim at providing integrity verification for different data storage systems, the problem of supporting both public auditability and data dynamics has not been fully addressed. How to achieve a secure and efficient design to seamlessly integrate these two important components for data storage service remains an open challenging task in Cloud Computing. Portions of the work presented in this paper have previously appeared as an extended abstract [1]. We revise the paper a lot and add more technical details as compared to [1].

3 Problem Statement

3.1 System Model

A representative network architecture for cloud data storage is illustrated in Fig. 1. Three different network entities can be identified as follows:

Client: an entity, which has large data files to be stored in the cloud and relies on the cloud for data maintenance and computation, can be either individual consumers or organizations;

Cloud Storage Server (CSS): an entity, which is managed by Cloud Service Provider (CSP), has significant storage space and computation resource to maintain the clients' data;

Third Party Auditor: an entity, which has expertise and capabilities that clients do not have, is trusted to assess and expose risk of cloud storage services on behalf of the clients upon request.

In the cloud paradigm, by putting the large data files on the remote servers, the clients can be relieved of the burden of storage and computation. As clients no longer possess their data locally, it is of critical importance for the clients to ensure that their data are being correctly stored and maintained. That is, clients should be equipped with certain security means so that they can periodically verify the correctness of the remote data even without the existence of local copies.

In case that clients do not necessarily have the time, feasibility or resources to monitor their data, they can delegate the monitoring task to a trusted TPA.

2.2 Security Model

Following the security model defined in [4], we say that the checking scheme is secure if 1) there exists no polynomialtime algorithm that can cheat the verifier with nonnegligible probability; and 2) there exists a polynomialtime extractor that can recover the original data files by carrying out multiple challenges-responses. The client or TPA can periodically challenge the storage server to ensure the correctness of the cloud data, and the original files can be recovered by interacting with the server. The authors in [4] also define the correctness and soundness of their scheme: the scheme is correct if the verification algorithm accepts when interacting with the valid prover (e.g., the server returns a valid response) and it is sound if any cheating server that convinces the client it is storing the data file is actually storing that file. Note that in the "game" between the adversary and the client, the adversary has full access to the information stored in the server, i.e., the adversary can play the part of the prover (server). The goal of the adversary is to cheat the verifier successfully, i.e., trying to generate valid responses and pass the data verification without being detected.

Our security model has subtle but crucial difference from that of the existing PDP or PoR models [2], [3], [4] in the verification process. As mentioned above, these schemes do not consider dynamic data operations, and the block insertion cannot be supported at all. This is because the construction of the signatures is involved with the file index information .HðnamekiÞ or hðvkiÞ should be generated by the client in the verification process. However, in our new construction the client has no capability to calculate HðmiÞ without the data information. In order to achieve this blockless verification, the server should take over the job of computing HðmiÞ and then return it to the prover. The consequence of this variance will lead to a serious problem: it will give the adversary more opportunities to cheat the prover by manipulating HðmiÞ or mi.

Due to this construction, our security model differs from that of the PDP or PoR models in both the verification and the data updating process. Specifically, the tags in our scheme should be authenticated in each protocol execution other than calculated or prestored by the verifier (the details will Fig. 1. Cloud data storage architecture. be shown in Section 3). In the following descriptions, we will use server and prover (or client, TPA, and verifier) interchangeably.

3.3 Design Goals

Our design goals can be summarized as the following:

1. Public auditability for storage correctness assurance to allow anyone, not just the clients who originally stored the file on cloud servers, to have the capability to verify the correctness of the stored data on demand.
2. Dynamic data operation support: to allow the clients to perform block-level operations on the data files while maintaining the same level of data correctness assurance. The design should be as efficient as possible so as to ensure the seamless integration of public auditability and dynamic data operation support.
3. Blockless verification: no challenged file blocks should be retrieved by the verifier (e.g., TPA) during verification process for efficiency concern.

4. System Analysis

Existing System:

From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons.

1.Firstly, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted due to the users' loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging.

2.Secondly, Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance.

Disadvantages:

These techniques, while can be useful to ensure the storage correctness without having users possessing data, can not address all the security threats in cloud data storage, since they are all focusing on single server scenario and most of them do not consider dynamic data operations.

As an complementary approach, researchers have also proposed distributed protocols for ensuring storage correctness across multiple servers or peers. Again, none of these distributed schemes is aware of dynamic data operations. As a result, their applicability in cloud data storage can be drastically limited.

Proposed System:

In this paper, we propose an effective and flexible distributed scheme with explicit dynamic data support to ensure the correctness of users' data in the cloud. We rely on erasure correcting code in the file distribution preparation to provide redundancies and guarantee the data dependability. This construction drastically reduces the communication and storage overhead as compared to the traditional replication-based file distribution techniques. By utilizing the homomorphic token with distributed verification of erasure-coded data, our scheme achieves the storage correctness insurance as well as data error localization: whenever data corruption has been detected during the storage correctness verification, our scheme can almost guarantee the simultaneous localization of data errors, i.e., the identification of the misbehaving server(s).

Advantages:

1. Compared to many of its predecessors, which only provide binary results about the storage state across the distributed servers, the challenge-response protocol in our work further provides the localization of data error.

2. Unlike most prior works for ensuring remote data integrity, the new scheme supports secure and efficient dynamic operations on data blocks, including: update, delete and append.

3. Extensive security and performance analysis shows that the proposed scheme is highly efficient and resilient against Byzantine failure, malicious data modification attack, and even server colluding attacks.

5. Security Analysis

In this section, we evaluate the security of the proposed scheme under the security model Following [4], we consider a file F after Reed-Solomon coding.

Definition 1 (CDH Problem):

CDH assumption holds in G if no t time algorithm has the non-negligible probability in solving the CDH problem. A proof-of-retrievability protocol is sound if any cheating prover that convinces the verification algorithm that it is storing a file F is actually storing that file, which we define in saying that it yields up the file F to an extractor algorithm which interacts with it using the proof-of-retrievability protocol. We say that the adversary (cheating server) is admissible if it convincingly answers an fraction of verification challenges. We formalize the notion of an extractor and then give a precise definition for soundness.

Theorem1. If the signature scheme is existentially unforgeable and the computational Diffie-Hellman problem is hard in bilinear groups, no adversary against the soundness of our public verification scheme could cause verifier to accept in a proof-of retrievability protocol instance with non-negligible probability, except by responding with correctly computed values.

Theorem 2. Suppose a cheating prover on an n-block file F is well-behaved in the sense above, and that it is admissible. Proof. The verification of the proof-of-retrievability is similar to [4], we omit the details of the proof here. The difference in our work is to replace HðiÞ with HðmiÞ such that secure update can still be realized without including the index information. These two types of tags are used for the same purpose (i.e., to prevent potential attacks), so this change will not affect the extraction algorithm defined in the proof-of-retrievability. We can also prove that extraction always succeeds against a well-behaved cheating prover, with the same probability analysis given in [4].

Theorem 3. Given a fraction of the n blocks of an encoded file F, it is possible to recover the entire original file F with all but negligible probability. Proof. Based on the rate Reed-Solomon codes, this result can be easily derived, since any fraction of encoded fileblocks suffices for decoding.

The security proof for the multiclient batch auditing is similar to the single-client case, thus omitted here.

6. Performance Analysis

We list the features of our proposed scheme in Table 3 and make a comparison of our scheme and the state of the art. The scheme in [14] extends the original PDP [2] to support data dynamics using authenticated skip list. Thus, we call it DPDP scheme thereafter. For the sake of completeness, we implemented both our BLS and RSA-based instantiations as well as the state-of-the-art scheme [14] in Linux.

Our experiment is conducted using C on a system with an Intel Core 2 processor running at 2.4 GHz, 768 MB RAM, and a 7200 RPM Western Digital 250 GB Serial ATA drive with an 8 MB buffer. Algorithms (pairing, SHA1 etc.) are implemented using the Pairing-Based Cryptography (PBC) library version 0.4.18 and the crypto library of OpenSSL version 0.9.8h. To achieve 80-bit security parameter, the curve group we work on has a 160-bit group order and the size of modulus N is 1,024 bits. All results are the averages of 10 trials. Table 4 lists the performance metrics for 1 GB file under various erasure code rate while maintaining high detection probability (99 percent) of file corruption.

From Table 4, it can be observed that the overall performance of the three schemes are comparable to each other. Due to the smaller block size (i.e., 20 bytes) compared to RSA-based instantiation, our BLS-based instantiation is more than two times faster than the other two in terms of server computation time. However, it has larger computation cost at the verifier side as the pairing operation in BLS scheme consumes more time than RSA techniques. Notethat the communication cost of DPDP scheme is the largest among the three in practice. This is because there are 4 tuple values associated with each skip list node for one proof, which results in extra communication cost as compared to our constructions. The communication overhead (server's response to the challenge) of our RSA-based instantiation and DPDP scheme [14] under different block sizes is illustrated in Fig. 6 We also conduct experiments for multiclient batch auditing and demonstrate its efficiency in Fig. 7, where the number of clients in the system is increased from 1 to approximately 100 with intervals of 4. As we can see, batch auditing not only enables simultaneously verification from multiple-client, but also reduces the computation cost on the TPA side. Given total K clients in the system, the batch auditing equation helps reduce the number of expensive pairing operations from 2K, as required in the individual auditing.

	Our BLS-based instantiation		Our RSA-based instantiation		[14]
Metric \ Rate-ρ	99%	97%	99%	97%	99%
Sever comp. time (ms)	6.45	2.11	13.81	4.55	14.13
Verifier comp. time (ms)	806.01	284.17	779.10	210.47	782.56
Comm. cost (KB)	239	80	223	76	280

Fig 2:Performance Comparison under Different Tolerance Rate of File Corruption for 1GB File

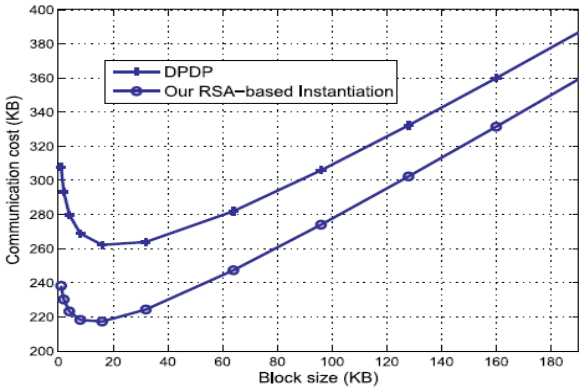
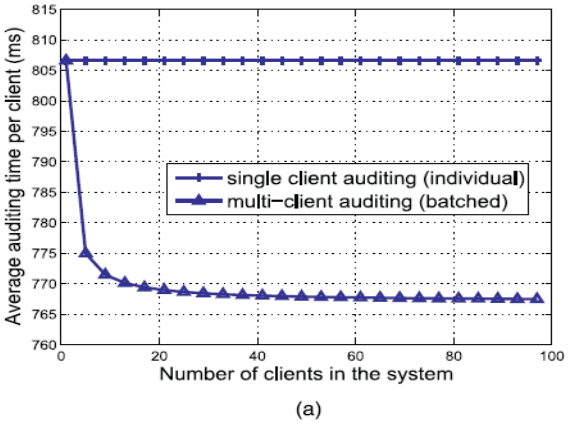
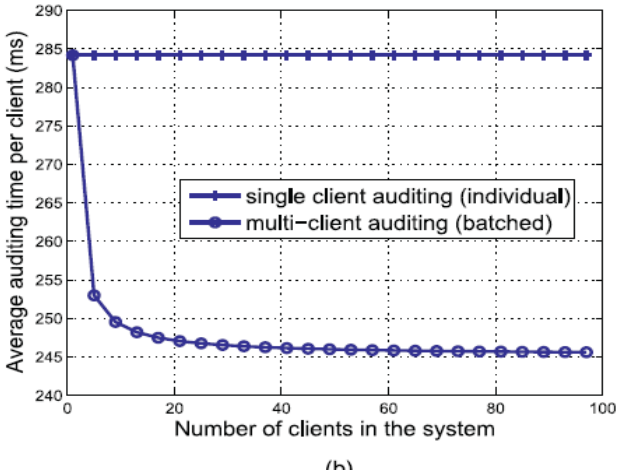


Fig 3: Comparison of communication complexity between our RSAbased instantiation and DPDP [14], for 1 GB file with variable block sizes. The detection probability is maintained to be 99 percent.



(a)



(b)

Fig 4: Performance comparison between individual auditing and batch auditing. The average per client auditing time is computed by dividing total auditing time by the number of clients in the system.

7. Results And Discussion

The below fig shows the services in the cloud.

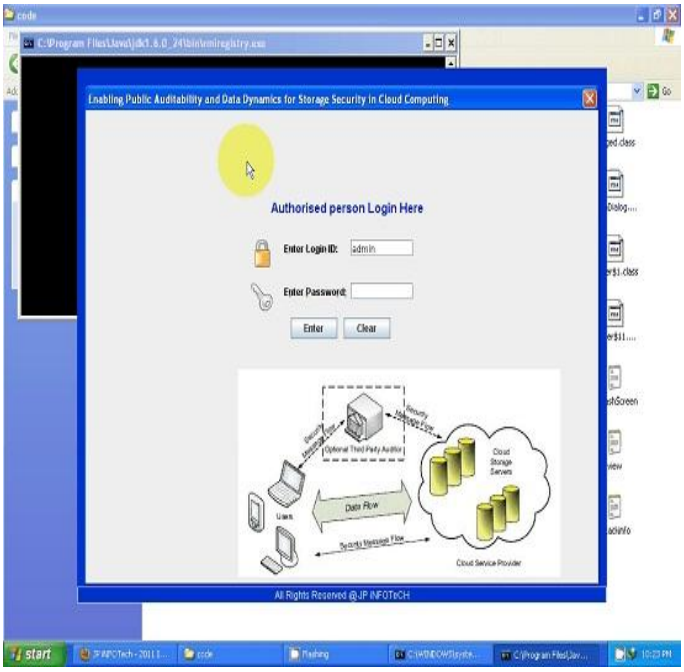


Fig 5: Cloud Login page

Fig 6:Cloud Authentication server

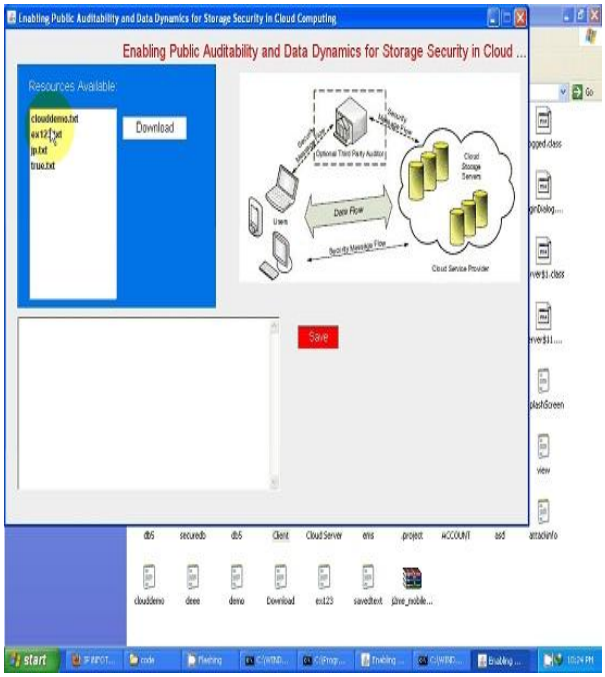


Fig 7:Resource page

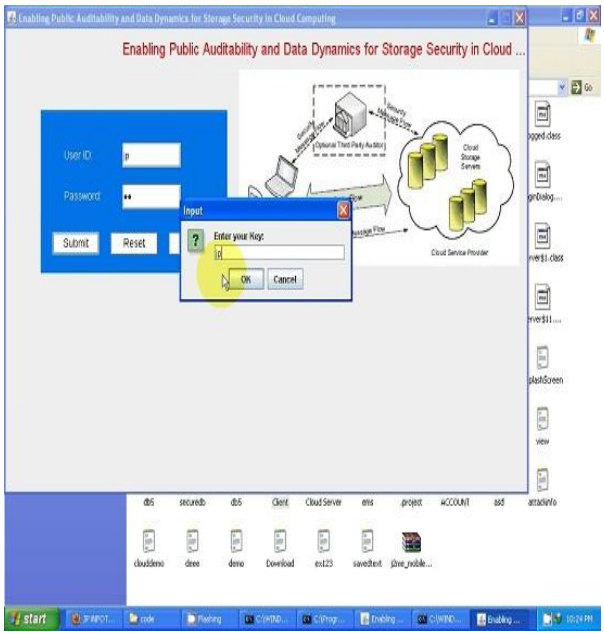


Fig 8:User Authentication by entering key

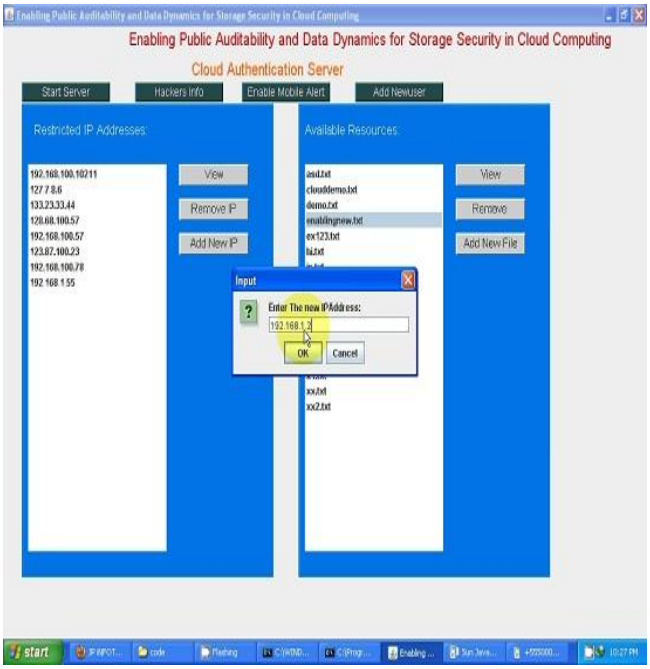


Fig 9: IP address of user

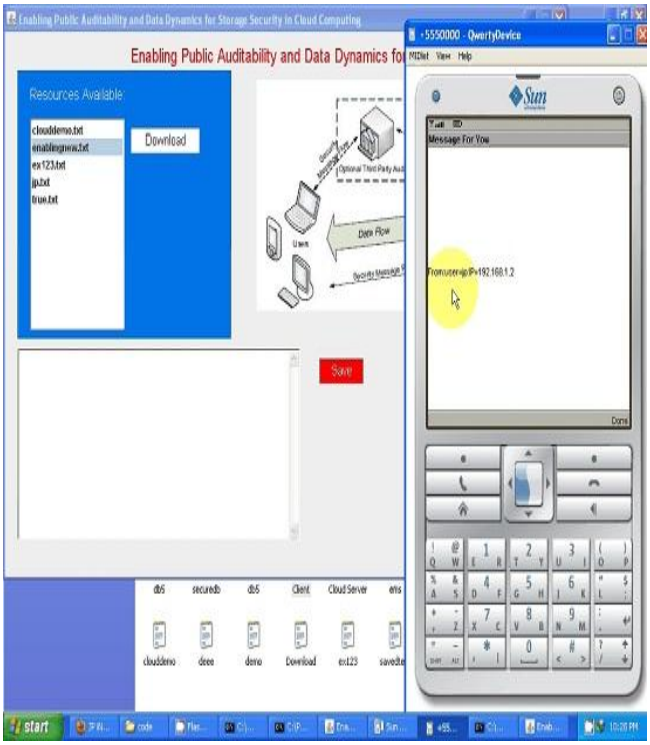


Fig 10:Mobile Alert page

CONCLUSION

In this paper, we investigated the problem of data security in cloud data storage, which is essentially a distributed storage system. To ensure the correctness of users' data in cloud data storage, we proposed an effective and flexible distributed scheme with explicit dynamic data support, including block update, delete, and append. We rely on erasure-correcting code in the file distribution preparation to provide redundancy parity vectors and guarantee the data dependability. By utilizing the homomorphic token with distributed verification of erasure coded data, our scheme achieves the integration of storage correctness insurance and data error localization, i.e., whenever data corruption has been detected during the storage correctness verification across the distributed servers, we can almost guarantee the simultaneous identification of the misbehaving server(s). Through detailed security and performance analysis, we show that 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 paramount importance, is still in its infancy now, and many research problems are yet to be identified. We envision several possible directions for future research on this area.

REFERENCES

[1] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamics for Storage Security in CloudComputing," Proc. 14th European Symp. Research in Computer Security (ESORICS '09), pp. 355-370, 2009.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 598-609, 2007.

[3] A. Juels and B.S. Kaliski Jr., "Pors: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), pp. 584-597, 2007.

[4] H. Shacham and B. Waters, "Compact Proofs of Retrievability," Proc. 14th Int'l Conf. Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08), pp. 90-107, 2008.

[5] K.D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Report 2008/175, Cryptology ePrint Archive, 2008.

[6] M. Naor and G.N. Rothblum, "The Complexity of Online Memory Checking," Proc. 46th Ann. IEEE Symp. Foundations of Computer Science (FOCS '05), pp. 573-584, 2005.

[7] E.-C. Chang and J. Xu, "Remote Integrity Check with Dishonest Storage Server," Proc. 13th European Symp. Research in Computer Security (ESORICS '08), pp. 223-237, 2008.

[8] M.A. Shah, R. Swaminathan, and M. Baker, "Privacy-Preserving Audit and Extraction of Digital Contents," Report 2008/186, Cryptology ePrint Archive, 2008.

[9] A. Oprea, M.K. Reiter, and K. Yang, "Space-Efficient Block Storage Integrity," Proc. 12th Ann. Network and Distributed System Security
Symp. (NDSS '05), 2005.

[10] T. Schwarz and E.L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," Proc. 26th IEEE Int'l Conf. Distributed Computing Systems (ICDCS '06), p. 12, 2006.

[11] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance," Proc. IEEE INFOCOM, pp. 954-962, Apr. 2009.

[12] G. Ateniese, R.D. Pietro, L.V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," Proc. Fourth Int'l Conf. Security and Privacy in Comm. Networks (SecureComm '08), pp. 1-10, 2008.

[13] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data StorageSecurity in Cloud Computing," Proc. 17th Int'l Workshop Quality of Service (IWQoS '09), 2009.

[14] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), 2009.

[15] K.D. Bowers, A. Juels, and A. Oprea, "Hail: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09), pp. 187-198, 2009.