International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 06 | June - 2022 Impact Factor: 7.185 ISSN: 2582-3930

Decentralization Access Control with Anonymous Authentication of Data Stored Using Game Theory

V.BRINDHAVANM
RIYA RAJ
J.SUHAINA FATHIMA
DHAANISH AHMED COLLEGE OF ENGINEERING

Abstract-The rapid development of the Internet, cloud storage has penetrated into every aspect of human society. However, cloud data disclosure happens more and more frequently, which makes cloud data security and privacy protection impact wide adoption of cloud storage. Control cloud data access based on reputation by introducing a Reputation Center (RC) was proposed and demonstrated to secure cloud data effectively in [9]. But the acceptance of such a system by cloud users and Cloud Service Providers (CSPs) is crucial for its practical deployment and final success.In this paper, we investigate the acceptance of a cloud data access control system based on reputation using Game Theory. Due to the existence of dishonest CSPs, there exists a social reputation dilemma among CSPs, which seriously impedes the popularity of cloud storage.To encourage users to use cloud storage and suppress collusion between CSPs and data requesters, a repeated public-goods game is built up by applying a compensation mechanism to improve the utilities of cloud users and a punishment mechanism based on reputation to incent honest behaviors.

1.INTRODUCTION

Cloud computing is a technology that uses the internet and central remote servers to maintain data and applications. Cloud computing allows consumers and businesses to use applications without installation and access their personal files at any computer with internet access. This technology allows for much more efficient computing by centralizing storage, memory, processing and bandwidth. Cloud computing is a comprehensive solution that delivers IT as a service. The flexibility of cloud computing is a function of the allocation of resources on demand. Before cloud computing, websites and server-based applications were executed on a specific system. Cloud computing is broken down into three segments application, storage and connectivity Cloud Applications or Software as a Service (SaaS) refers to software delivered over a browser. SaaS eliminates the need to install and run applications on the customer's own computers/servers and simplifies maintenance, upgrades and support. Examples of SaaS are Facebook, SalesForce, etc.

EXISTING SYSTEM

The cloud infrastructures are much more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machine) and external threats (e.g., via system holes) that can damage data integrity, second, for the benefits of possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users. Occasionally suffer from the lack of trust on CSP because the data change may not be timely known by the cloud users, even if these disputes may result from the users' own improper operations. Security audit is an important solution enabling trace back and analysis of any activities including data accesses, security breaches, and application activities. Data security tracking is crucial for all organizations that should comply with a wide range of federal regulations. Compared to the common audit, the audit services for cloud storages should provide clients with a more efficient proof for verifying the integrity of stored data. Unfortunately, the traditional cryptographic technologies, based on hash functions and signature schemes, cannot support for data integrity verificatio

EXISTING SYSTEM

The cloud infrastructures are much more powerful and reliable than personal computing devices, but they are still susceptible to internal threats (e.g., via virtual machine) and external threats (e.g., via system holes) that can damage data integrity, second, for the benefits of possession, there exist various motivations for cloud service providers (CSP) to behave unfaithfully toward the cloud users.Occasionally suffer from the lack of trust on CSP because the data change may not be timely known by the cloud users, even if these disputes may result from the users' own improper operations. Security audit is an important solution enabling trace back and analysis of any activities including data accesses, security breaches, and application activities. Data security tracking is crucial for all organizations that should comply with a wide range of federal regulations. Compared to the common audit, the audit services for cloud storages should provide clients with a more efficient proof for verifying the integrity of stored data. Unfortunately, the traditional cryptographic technologies, based on hash functions and signature schemes, cannot support for data integrity verification without a local copy of data. In addition, it is evidently impractical for audit services to download the whole data for checking data validation due to the communication cost, especially for large-size files.

PROPOSED SYSTEM

Introduce a dynamic audit service for integrity verification of un-trusted and outsourced storages. These audit service can provide public audit ability without downloading raw data and protect privacy of the data. Also, this audit system can support dynamic data operations and timely anomaly detection with the help of several effective techniques, such as **fragment structure**, **random sampling**, and I**ndex-Hash Table (IHT),** also propose an efficient approach based on **probabilistic query** and **periodic**

verification for improving the performance of audit services. Another major concern is the security issue of dynamic data operations for public audit services. In clouds, one of the core design principles is to provide dynamic scalability for various applications. This means that remotely stored data might be not only accessed by the clients but also dynamically updated by them, for instance, through block operations such as **modification, deletion** and **insertion**.

SYSTEM SPECIFICATION

SOFTWARE REQUIREMENT

Front End/GUI Tool	: Microsoft Visual studio 2010 and above
Operating System	: Windows Family 7 or 8
Language	: C#
Application	: Web Application (ASP.NET)
Back end	: Windows Azure Storage
Developer SDK	: Windows Azure SDK 2.2

HARDWARE REQUIREMENT

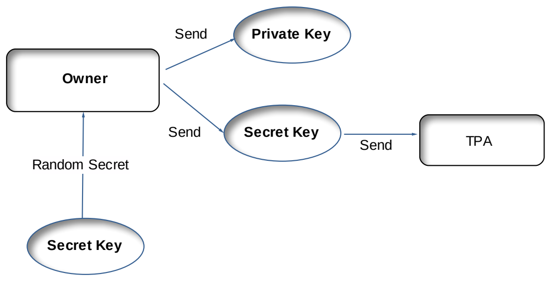
Processor	: Pentium dual core
RAM	: 1 GB
Hard Disk Drive	: 80 GB
Monitor	: 17" Color Monitor

MODULE DESCRIPTION

Key Generation:

The owner generates a public/secret key pair (pk, sk) by himself or the system manager, and then sends his public key pk to TPA. Note that TPA cannot obtain the client's secret key sk; secondly, the owner chooses the random secret.

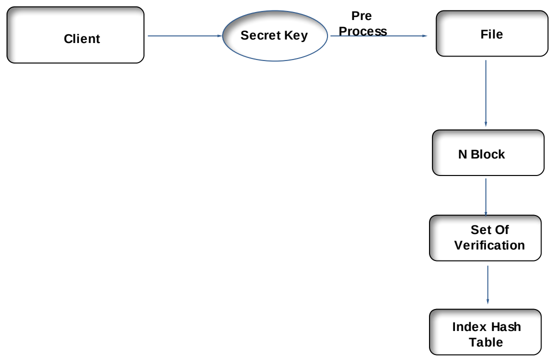
The key feels like, should do the right thing with their cloud strategy and make sure that they ask the right questions to their cloud service providers. It isn't really about whether or not the service has encryption or not.



Tag Generation:

The client (data owner) uses the secret key sk to pre-process a file, which consists of a collection of n blocks, generates a set of public verification parameters and index-hash table that are stored in TPA, and transmits the file and some verification tags to CSP.

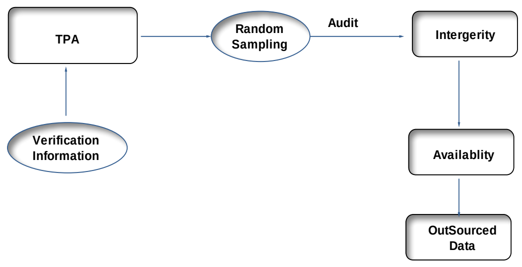
Tags generated by DOs and the leakage of the user's secret key



Periodic Sampling Audit:

1.TPA (or other applications) issues a “Random Sampling” challenge to audit the integrity and availability of outsourced data in terms of the verification information stored in TPA.

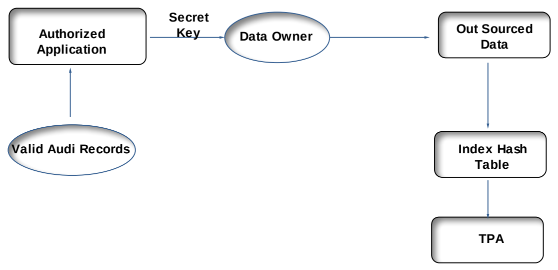
2 .It is crucial to develop a more efficient and secure mechanism for dynamic audit services, in which a potential adversary’s advantage through dynamic data operations



Audit for Dynamic Operations:

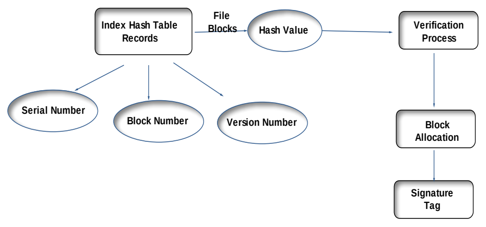
An authorized application, which holds data owner’s secret key sk, can manipulate the outsourced data and update the associated index hash table stored in TPA. The privacy of sk and the checking algorithm ensure that the storage server cannot cheat the authorized applications and forge the valid audit records.

AAs should be cloud application services inside clouds for various application purposes, but they must be specifically authorized by DOs for manipulating outsourced data. Since the acceptable operations require that the AAs must present authentication information for TPA, any unauthorized modifications for data will be detected in audit processes or verification processes. Based on this kind of strong authorization-verification mechanism, we assume neither CSP is trusted to guarantee the security of stored data, nor a DO has the capability to collect the evidence of CSP’s faults after errors have been found.



Index hash map:

To support dynamic data operations, we introduce a simple IHT to record the changes of file blocks, as well as generate the hash value of each block in the verification process. The structure of our IHT is similar to that of file Block allocation table in file systems. Generally, the IHT consists of serial number, block number, version number, and random integer Note that we must assure all records in the IHT differ from one another to prevent the forgery of data blocks and tags. In addition to recording data changes, each record _i in the table is used to generate a unique hash value, which in turn is used for the construction of a signature tag _i by the secret key sk. The relationship between _i and _i must b e crypto graphically secure, and we make use of it to design our verification protocol.



LITERATURE SURVEY

1PORs: Proofs of Retrievability for Large Files

Cloud computing promises In this paper, we define and explore *proofs of retrievability* (PORs). A POR scheme enables an archive or back-up service (prover) to produce a concise proof that a user (verifier) can retrieve a target file F, that is, that the archive retains and reliably transmits file data sufficient for the user to recover F in its entirety.

A POR may be viewed as a kind of cryptographic proof of knowledge (POK), but one specially designed to handle a *large* file (or bitstring) F. We explore POR protocols here in which the communication costs, number of memory accesses for the prover, and storage requirements of the user (verifier) are small parameters essentially independent of the length of F. In addition to proposing new, practical POR constructions, we explore implementation considerations and optimizations that bear on previously explored, related schemes. In a POR, unlike a POK, neither the prover nor the verifier need actually have knowledge of F. PORs give rise to a new and unusual security definition whose formulation is another contribution of our work. We view PORs as an important tool for semi-trusted online archives. Existing cryptographic techniques help users ensure the privacy and integrity of files they retrieve. It is also natural, however, for users to want to verify that archives do not delete or modify files prior to retrieval. The goal of a POR is to accomplish these checks *without users having to download the files themselves*. A POR can also provide quality-of-service guarantees, i.e., show that a file is retrievable within a certain time bound.

Drawbacks:

To illustrate the basic idea and operation of a POR, it is worth considering a straightforward design involving a keyed hash function h_(F). In this scheme, prior to archiving a file F, the verifier computes and stores a hash value r = h_(F) along with secret, random key _. To check that the prover possesses F, the verifier releases _ and asks the prover to compute and return r. Provided that h is resistant to second-preimage attacks, this simple protocol provides a strong proof that the prover knows F. By storing multiple hash values over different keys, the verifier can i nitiate multiple, independent checks. This keyed-hash approach, however, has an important drawback: High resource costs. The keyedhash protocol requires that the verifier store a number of hash values linear in the number of checks it is to perform. This characteristic conflicts with the aim of enabling the verifier to offload its storage burden. More significantly, each protocol invocation requires that the prover process the *entire* file F. For large F, even a computationally lightweight operation like hashing can be highly burdensome. Furthermore, it requires that the prover read the entire file for every proof—a significant overhead for an archive whose intended load is only an occasional read per file, were every file to be tested frequently.

Contribution:

We introduce a POR protocol in which the verifier stores only a single cryptographic key—irrespective of the size and number of the files whose retrievability it seeks to verify—as well as a small amount of dynamic state (some tens of bits) for each file. (One simple variant of our protocol allows for the storage of no dynamic state, but yields weaker security.) More strikingly, and somewhat counterintuitively, our scheme requires that the prover access only a small portion of a (large) file F in the course of a POR. In fact, the portion of F "touched" by the prover is essentially independent of the length of F and would, in a typical parameterization, include just hundreds or thousands of data

blocks. Briefly, our POR protocol encrypts F and randomly embeds a set of randomly-valued check blocks called *sentinels*. The use of encryption here renders the sentinels indistinguishable from other file blocks. The verifier challenges the prover by specifying the positions of a collection of sentinels and asking the prover to return the associated sentinel values. If the prover has modified or deleted a *substantial* portion of F, then with high probability it will also have suppressed a number of sentinels. It is therefore unlikely to respond correctly to the verifier. To protect against corruption by the prover of a *small* portion of F, we also employ error-correcting codes. We let ˜ F refer to the full, encoded file stored with the prover. A drawback of our proposed POR scheme is the preprocessing / encoding of F required prior to storage with the prover. This step imposes some computational overhead—beyond that of simple encryption or hashing—as well as larger storage requirements on the prover. The sentinels may constitute a small fraction of the encoded ˜ F (typically, say, 2%); the error-coding imposes the bulk of the storage overhead. For large files and practical protocol parameterizations, however, the associated expansion factor | ˜ F|/|F| can be fairly modest, e.g., 15%.

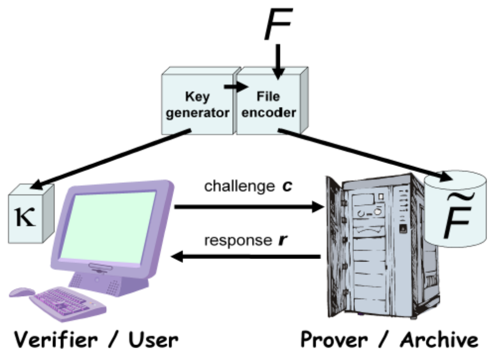
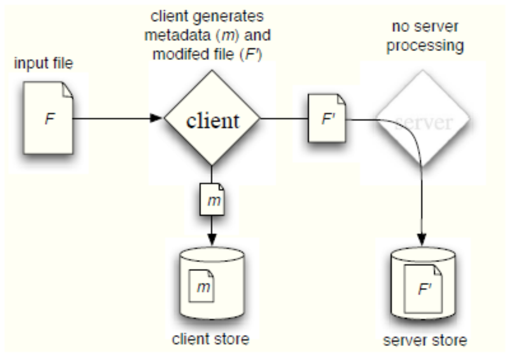


Fig 2: Schematic of a POR system. An encoding algorithm transforms a raw file F into an encoded file ˜ F to be stored with the prover / archive. A key generation algorithm produces a key _ stored by the verifier and used in encoding. (The key _ is independent of F in some PORs, as in our main scheme.) The verifier performs a challenge-response protocol with the prover to check that the verifier can retrieve F.

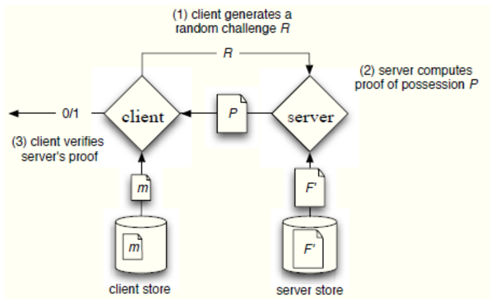
2.2 Provable data possession at untrusted stores

Introduce a model for provable data possession (PDP) that allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O costs. The client maintains a constant amount of metadata to verify the proof. The challenge/response protocol transmits a small, constant amount of data, which minimizes network communication. Thus, the PDP model for remote data checking supports large data sets in widely-distributed storage system.

We present two provably-secure PDP schemes that are more efficient than previous solutions, even when compared with schemes that achieve weaker guarantees. In particular, the overhead at the server is low (or even constant), as opposed to linear in the size of the data. Experiments using our implementation verify the practicality of PDP and reveal that the performance of PDP is bounded by disk I/O and not by cryptographic computation.



(a) Pre-process and store


(1) client generates a random challenge R
R
(2) server computes proof of possession P
0/1
client
P
server
(3) client verifies server's proof
m
F'
m
F'
client store
server store
(b) Verify server possession

Drawbacks:

However, archival storage requires guarantees about the authenticity of data on storage, namely that storage servers possess data. It is insufficient to detect that data have been modified or deleted when accessing the data, because it may be too late to recover lost or damaged data. Archival storage servers retain tremendous amounts of data.

Contributions:

In this paper we:

- Formally define protocols for provable data possession (PDP) that provide probabilistic proof that a third party stores a file.
- Introduce the first provably-secure and practical PDP schemes that guarantee data possession.
- Implement one of our PDP schemes and show experimentally that probabilistic possession guarantees make it practical to verify possession of large data sets.

Our PDP schemes provide data format independence, which is a relevant feature in practical deployments, and put no restriction on the number of times the client can challenge the server to prove data possession. Also, a variant of our main PDP scheme offers public verifiability.

2.3 Demonstrating data possession and un-cheatable data transfer

We observe that a certain RSA-based secure hash function is homomorphic. We describe a protocol based on this hash function which prevents ‘cheating’ in a data transfer transaction, while placing little burden on the trusted third party that oversees the protocol. We also describe a cryptographic protocol based on similar principles, through which a prover can demonstrate possession of an arbitrary set of data known to the verifier. The verifier isn’t required to have this data at hand during the protocol execution, but rather only a small hash of it. The protocol is also provably as secure as integer factoring. this raises many concerns over the honesty of network users. More explicitly, it is desirable to know, in a distributed data store network, whether users are actually storing files they were assigned to store; and in a content distribution network where users are rewarded for donating their idle bandwidth and charged for using other users' bandwidth, whether data transfers took place correctly and bandwidth credits for the transaction can be exchanged. We cite some scenarios in Sections 3 and 4 where it would make sense for a user to break network rules in exchange for some form of personal gain (or plain vandalism). Hence it is desirable to keep tabs on the honesty of users, which is the purpose of the protocols described in this paper.

Drawbacks:

It is interesting to note that a personal computer has three main resources which may be exploited by a distributed network: processing time, network bandwidth and storage.

In each case a user may 'cheat' and not dedicate the resources as promised. For the first, efficient techniques are known for certain classes of problems which detect with arbitrarily high accuracy any attempt to shortcut computations [4]. In this paper, we provide limited (yet still useful) ways to prevent cheating when the latter two resources are involved.

Contribution:

A function H is homomorphic if, given two operations + and ×, we have

$H(d + d') = H(d) \times H(d')$.

An homomorphic hash function is, simply put, a hash function that is homomorphic.

In many cases it is undesirable that a hash function be homomorphic, and most known constructs of this type are weak. However, it is possible to build strong homomorphic hash functions based on public-key primitives, so long as the secret parameters are not disclosed; up until now, the only known example was the work of Krohn, Freedman and Mazires [6], based on discrete logarithms. We now describe a different homomorphic hash function, based on principles similar to RSA, and which is slightly more flexible than Krohn-Freedman-Mazires's function. This function isn't novel; see e.g. [10]. However, we believe nobody has yet called to attention its homomorphic property.

2.4 Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing

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 multiuser 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.

Drawbacks:

The first to explore constructions for dynamic provable data possession, They extend the PDP model 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 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.

Contributions:

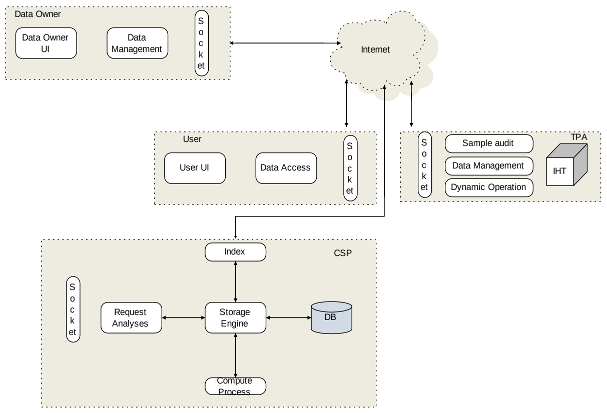
Our contribution can be summarized as follows:

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.

ARCHITECTURE DIAGRAM



ALGORITHM

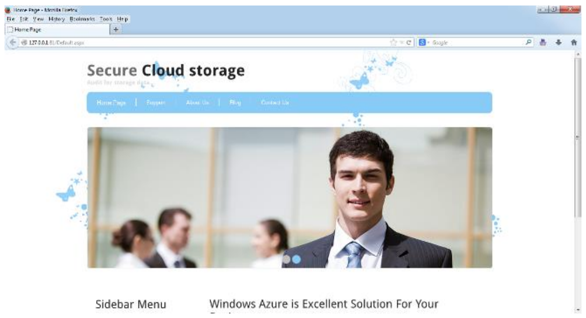
INDEX HASHING TABLE ALGORITHM

Hashing is a technique or process of mapping keys, and values into the hash table by using a hash function. It is done for faster access to elements. The efficiency of mapping depends on the efficiency of the hash function used.

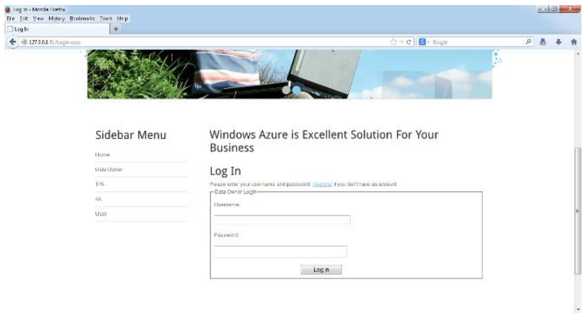
Let a hash function H(x) maps the value at the index **x%10** in an Array. For example if the list of values is [11,12,13,14,15] it will be stored at positions {1,2,3,4,5} in the array or Hash table respectively.

SCREENSHOTS

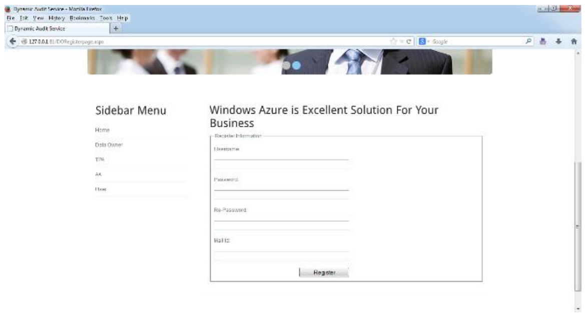
SCREEN SHOTS:



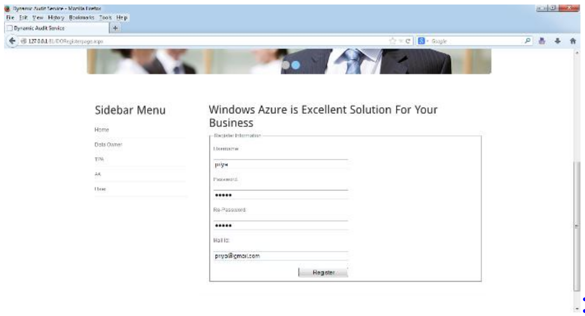
LOGIN PAGE



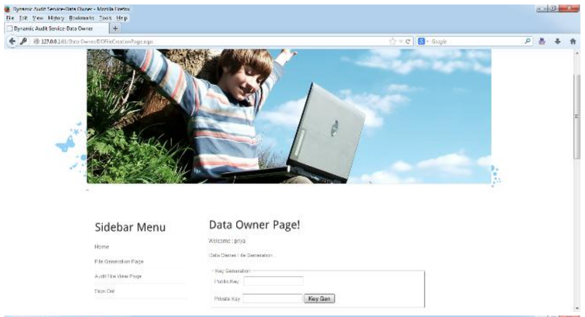
REGISTER PAGE:

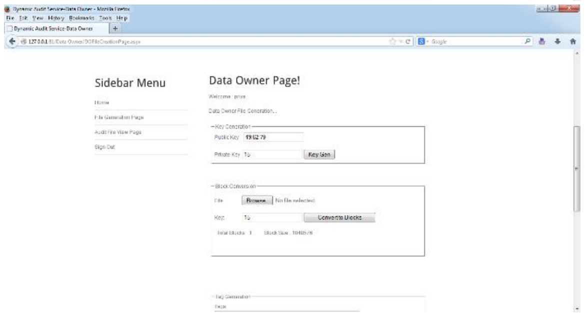


KEY GENERATION AND FILE BLOCK CONVERSION

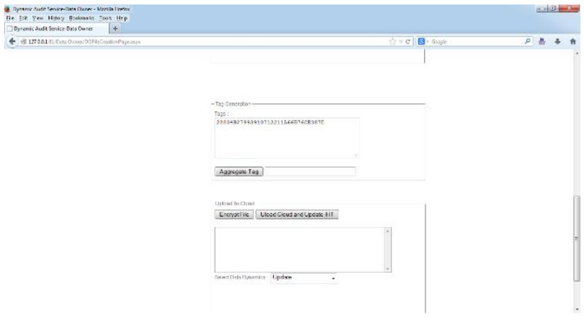

Windows Azure is Excellent Solution For Your Business

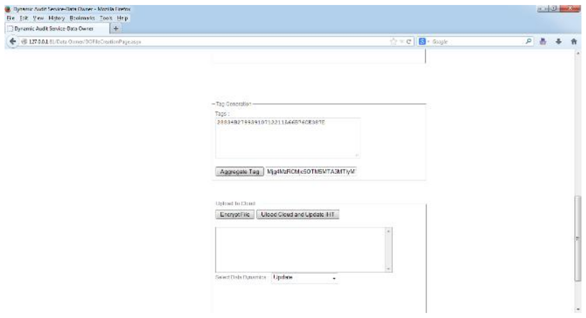
Register Information
Username: priya
Password: •••••
Re-Password: •••••
Mail Id: priya@gmail.com


Data Owner Page!
Welcome : priya
Data Owner File Generation..
Key Generation
Public Key:
Private Key:

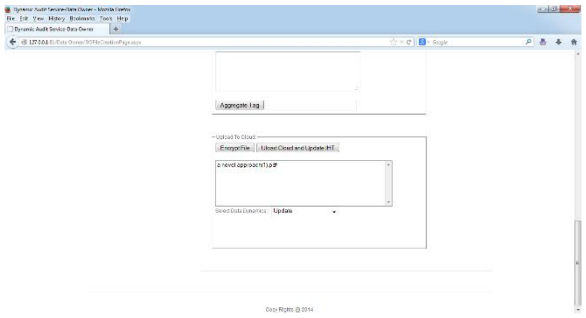

Data Owner Page!
Welcome : priya
Data Owner File Generation..
Key Generation
Public Key: 19.62.79
Private Key: 15
Block Conversion
File: No file selected
Key: 15
Total Blocks : 1 Block Size : 1048576
Tag Generation
Tags :

TAG GENERATION

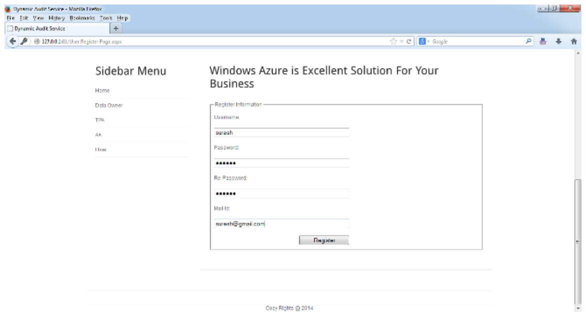

Tag Generation
Tags : 288348279939107122111A66B76CB387E
Upload In Cloud
Select Data Dynamics: Update


Tag Generation
Tags : 288348279939107122111A66B76CB387E
Aggregate Tag: Mjg4MzRCMzE5OTM5MTA3MTIyM
Upload In Cloud
Select Data Dynamics: Update

UPLOADED FILE

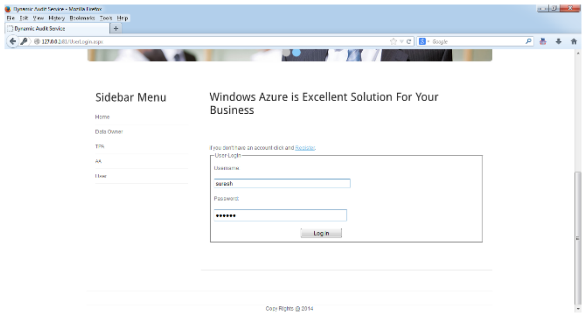

Upload In Cloud
a novel approach(1).pdf
Select Data Dynamics: Update
Copy Rights @ 2014

USER REGISTER PAGE


Windows Azure is Excellent Solution For Your Business

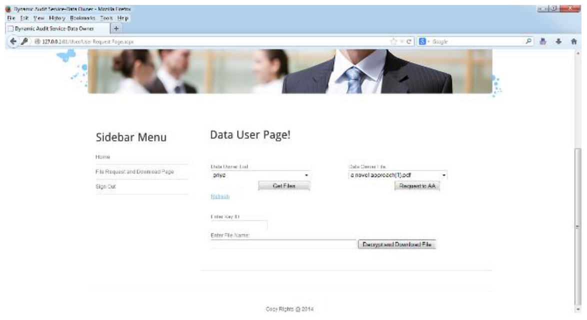
Register Information
Username: suresh
Password: ••••••
Re-Password: ••••••
Mail Id: suresh@gmail.com
Copy Rights @ 2014

USER LOGIN PAGE


Windows Azure is Excellent Solution For Your Business

If you don't have an account click and Register
User Login
Username: suresh
Password: ••••••
Copy Rights @ 2014

DATA USER PAGE


Dynamic Audit Service-Data Owner - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Dynamic Audit Service-Data Owner
127.0.0.1:81/User/User Request Page.aspx
Sidebar Menu
Home
File Request and Download Page
Sign Out
Data User Page!
Data Owner List
priya
Get Files
Data Owner File
a novel approach(1).pdf
Request to AA
Refresh
Enter Key ID
Enter File Name:
Decrypt and Download File
Copy Rights @ 2014

CONCLUSION

A construction of dynamic audit services for un-trusted and outsourced storages. We also presented an efficient method for periodic sampling audit to enhance the performance of TPAs and storage service providers. This project developed in Windows Azure Cloud, mainly this cloud improves the security and performance

FUTURE ENHANCEMENT

Hash Bucketing Based Knowledge module

A Bucketing hashing-based technique, called flexible Bucketing-based hashing, for processing the NN query. The main advantage of this technique is that the server always returns a exact binary candidate set. The client then refines the candidate set to obtain the final result.

REFERENCES

1. Amazon Web Services, "Amazon S3 Availability Event: July 20, 2008," http://status.aws.amazon.com/s3-20080720.html, July 2008.
2. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. ACM Conf. Computer and Communications Security (CCS '07), pp. 584-597, 2007.
3. M.Mowbray, "The Fog over the Grimpen Mire: Cloud Computing and the Law," Technical Report HPL-2009-99, HP Lab., 2009.
4. A.A.Yavuz and P.Ning, "BAF: An Efficient Publicly Verifiable Secure Audit Logging Scheme for Distributed Systems," Proc. Ann. Computer Security Applications Conf. (ACSAC), pp. 219-228, 2009.
5. G. Ateniese, R.C. Burns, R. Curtmola, J. Herring, L. Kissner, Z.N.J. Peterson, and D.X. Song, "Provable Data Possession at Untrusted Stores," Proc. 14th ACM Conf. Computer and Comm. Security, pp. 598-609, 2007.

1] Peter Mell, Tim Grance, et al. The nist definition of cloud computing. 2011.

[2] Yinghui Zhang, Robert H Deng, Shengmin Xu, Jianfei Sun, Qi Li, and Dong Zheng. Attribute-based encryption for cloud computing access control: A survey. ACM Computing Surveys, 53(4):1–41, 2020.

[3] Jia-Nan Liu, Xizhao Luo, Jian Weng, Anjia Yang, Xu An Wang, Ming Li, and Xiaodong Lin. Enabling efficient, secure and privacypreserving mobile cloud storage. IEEE Transactions on

Dependable and Secure Computing, pages 1–15, 2020.

[4] Anjia Yang, Jia Xu, Jian Weng, Jianying Zhou, and Duncan S Wong. Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage. IEEE Transactions on Cloud Computing, 9(1):212–225, 2021.

[5] Dawn Xiaoding Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pages 44–55. IEEE, 2000.

[6] Ming Zeng, Hai-Feng Qian, Jie Chen, and Kai Zhang. Forward secure public key encryption with keyword search for outsourced cloud storage. IEEE Transactions on Cloud Computing, pages 1–13, 2019.

[7] Yupeng Zhang, Jonathan Katz, and Charalampos Papamanthou. All your queries are belong to us: The power of file-injection attacks on searchable encryption. In 25th USENIX Security Symposium, pages 707–720, 2016.

[8] Emil Stefanov, Charalampos Papamanthou, and Elaine Shi. Practical dynamic searchable encryption with small leakage. In Network and Distributed System Security Symposium, volume 71, pages 72–75, 2014.

[9] Raphael Bost. Poφoς: Forward secure searchable encryption. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pages 1143–1154. ACM, 2016.

[10] Zhongjun Zhang, Jianfeng Wang, Yunling Wang, Yaping Su, and Xiaofeng Chen. Towards efficient verifiable forward secure searchable symmetric encryption. In European Symposium on Research in Computer Security, pages 304–321. Springer, 2019