ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Outsourcing Proofs of retrievability using cloud computing systems

G. Rajendra kumar
Department of Computer Science & Engineering
Jogaiah Institute of Technology and Sciences College of Engineering,
Palakol, West Godavari Dt., A.P-534 268, India.

Mr.A.Veerabhadra Rao
Department of Computer Science & Engineering
Jogaiah Institute of Technology and Sciences College of Engineering,
Palakol, West Godavari Dt., A.P-534 268, India.

ABSTRACT:

Proofs of retrievability (POR) are cryptographic proofs that enable a cloud provider to prove that a user can retrieve his file in its entirety. POR need to be frequently executed by the user to ensure that their files stored on the cloud can be fully retrieved at any point in time. To conduct and verify POR, users need to be equipped with devices that have network access, and that can tolerate the (non-negligible) computational overhead incurred by the verification process. This clearly hinders the large-scale adoption of POR by cloud users, since many users increasingly rely on portable devices that have limited computational capacity, or might not always have network access.

In this paper, we introduce the notion of outsourced proofs of retrievability (OPOR), in which users can task an external auditor to perform and verify POR with the cloud provider. We argue that the OPOR setting is subject to security risks that have not been covered by existing POR security models. To remedy that, we propose a formal framework and a security model for OPOR. We then propose an instantiation of OPOR which builds upon the provably-secure private POR scheme due to Sachem and Waters (Asiacrypt'08) and we show its security in our proposed security model. We implement a prototype based on our solution, and evaluate its performance in a realistic cloud setting. Our evaluation results show that our proposal minimizes user effort, incurs negligible overhead on the auditor (compared to the SW scheme), and considerably improves over existing publicly verifiable POR.

INTRODUCTION

Cloud Services:

Cloud services are increasingly gaining importance and applicability in numerous application domains, such as storage, computing services, collaboration platforms, etc. The success of the cloud model is driven by the tremendous economic benefit offered to companies, private individuals, and public

Cite this article as: K. Rajendra kumar & Mr.A.Veerabhadra Rao, " Outsourcing Proofs of retrievability using cloud computing systems", International Journal & Magazine of Engineering, Technology, Management and Research, Volume 6 Issue 8, 2019, Page 42-52.

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

Organizations to deploy/provision cloud services in a cost effective manner. The advent of cloud storage and computation services, however, introduces new threats to data security. As a matter of fact, customers of cloud services lose control over their data and how data is processed or stored. Indeed, this has been identified as the main obstacle which makes users reluctant when using cloud services .The literature features a number of solutions that enable users to verify the integrity and availability of their outsourced data .Examples include Proofs of retrievability (POR) which provide end-clients with the assurance that the data is still available and can be entirely downloaded if needed, and Proofs of Data Possession (PDP) which enable a client to verify that its stored data has not undergone any modifications, among others. All existing solutions share a similar system and attacker model, comprising of the cloud user and a rational cloud provider. Here, the 'malicious' cloud aims at minimizing storage costs, e.g., by not deploying the appropriate security measures in their datacenters, or by intentionally modifying (e.g., deleting) user data. Clearly, the guarantees provided by current solutions therefore largely depend on the users themselves who are required to regularly perform verifications (e.g., POR) in order to react as early as possible in case of data loss. Moreover, the verification of a POR requires the user to be equipped with devices that have network access, and that can tolerate the (non-negligible) computational overhead incurred by the verification process. Therefore, customers either have to (i) accept this burden and regularly verify their outsourced data (e.g., by invoking POR with the cloud provider), or (ii) entrust the cloud providers to deploy the necessary security mechanisms that ensure data integrity in spite of server failures, exploits, etc. We point out that the integration of such security.

Outsourced Proofs of Retrievability:

In this section, we introduce a formal model for OPOR. Since OPOR extends POR, we first introduce POR, adapted from Proofs of Retrievability Proofs of Retrievability (POR) are cryptographic proofs that prove the retrievability of outsourced data. More precisely, POR assume a model comprising of a user, and a service provider that stores a file pertaining to the user. POR consist basically of a challenge response protocol in which the service provider proves to the user that its file is still intact and retrievable. Note that POR only provide a guarantee that a fraction p of the file can be retrieved. For that reason, POR are typically performed on a file which has been erasure- coded in such a way that the recovery of any fraction p of the stored data ensures the recovery of the file. A POR scheme consists of four procedures, setup, store, verify, and prove: setup. This randomized algorithm generates the involved keys and distributes them to the parties. If public keys are involved, these are distributed amongst all parties. store. This randomized algorithm takes as input the keys of the user and a file $M \in \{0, 1\}*$. The file gets processed and it outputs the produced $M*$ which will be stored on the server. The algorithm also generates a file tag τ which contains additional information (e.g., metadata, secret information) about $M*$. verify, prove. The randomized proving and verifying algorithms define a protocol for proving file retrievability. We refer to this protocol as the POR protocol (in contrast to a POR scheme that comprises all four procedures). While the verifier algorithm takes the secret keys as input, the prover algorithm takes as input the processed file $M*$ that is output by store. Both verify, prove algorithms also take as input the public key and the file tag τ from store during protocol execution. Algorithm verify outputs at the end of the protocol run TRUE if the verification succeeds, meaning that the file is being stored on the server, and FALSE otherwise.

OPOR Model:

Similar to the traditional POR model, an OPOR consists of a user U, the data owner, who plans to outsource his data M to a service provider S. In addition, U is interested in acquiring regular proofs that his data is correctly stored and retrievable from S. To this end, an OPOR comprises a new entity A, called the auditor, who runs POR with S on behalf of U. If these POR do not succeed, the auditor takes certain actions, e.g., inform the user immediately. Otherwise, the user is assured that the data are stored correctly. More specifically, an OPOR scheme comprises five protocols Setup, Store, POR, CheckLog and ProveLog. The first three protocols resemble the protocols that are represented in a POR scheme (see Section 2.1) but extend them. One major difference is that the POR protocol not only outputs a decision on whether the POR has been correct, but also a log file. The log files serve a twofold purpose. First, they allow the user to check (using the CheckLog procedure) if the auditor did his job during the runtime of the OPOR scheme.

As the purpose of OPOR is to incur less burden on the user, the verification of the logs by the user should incur less resource consumption on the user when compared to the standard verification of POR directly with S. Second, logs allow the auditor to prove (using the ProveLog procedure) that if some problems occur, e.g., the file is no longer stored by S, the auditor must not be blamed. In what follows, we detail each protocol in OPOR. This randomized protocol generates for each of the different parties a public-private key pair. If a party only deploys symmetric key schemes, the public key is simply set to ⊥. For the sake of brevity, we implicitly assume for each of the subsequent protocols and procedures that an involved party always uses as inputs its own secret key and the public keys of the other parties. The Store Protocol. This randomized file-storing protocol takes the secret keys of the parties and a file M from the user to be stored. The output M∗ for the service provider marks the data that it should store. The user also needs a contract c specifying the policy for checks for the auditor. Observe that M∗ may not be exactly equal to M, but it must be guaranteed that M can be recovered from M∗ . Additionally, the output needs to contain information which (i) enables the execution of a POR protocol between A and S on the one hand and (ii) enables the validation of the log files created by A on the other hand. This information consists of two tokens represented by τA and τU , respectively.

The POR Protocol:

In the OPOR model, the auditor A and the provider S run a POR protocol to convince the auditor that M∗ is still retrievable from S. The input of A is the tag τA given by Store, and the input of the provider S is the stored copy of the file M∗ . Similar to the traditional POR model, on the auditor's side (who plays the role of the verifier), the output contains one binary value decA which expresses whether the auditor accepts the POR or not. In addition, the POR protocol will produce a log file Λ. It holds that: POR: [A : τA; S : M∗] −→ [A : Λ, decA]

The protocol run is accepted by the auditor if decA = TRUE.

Security Model:

In the following, we explain how security is defined within the OPOR model. We do not consider confidentiality of the file M, but assume that the user encrypts the file prior to the start the OPOR protocol. In OPOR, we extend the attacker model of traditional POR which only considers malicious service providers, and we assume that any subset of parties can be corrupted. To define the soundness of an OPOR scheme, we adapt and extend the existing POR security models. In security is formalized using the notion of an extractor algorithm, that is able to extract the file in interaction with the adversary. This proves the following statement: if the prover convinces the verifier with a sufficient level of probability then the file is actually stored. As already elaborated, the notion of extractability is not sufficient to capture security in OPOR schemes.

Fortress:

An Efficient OPOR In this section, we introduce and detail an efficient instantiation of OPOR. We analyze the security of our instantiation according to the model outlined. mechanisms in current clouds often incurs considerable costs on the cloud

Providers, which explains the reason why none of today’s cloud storage services accept liability for data loss in their Service Level Agreements (SLAs) and only guarantee service availability– in spite of the plethora of cloud security and dependability solutions that populate the literature.

OPOR protects against a malicious auditor, and malicious users/cloud providers (and against collusion among any combination of these parties); we contrast this to existing public (and delegable) POR ,In this paper, we address this problem, and propose a novel solution—outsourced proofs of retrievability (OPOR)—which goes one step beyond existing POR and enables an external party, the auditor, to execute a POR protocol with the cloud provider on behalf of the data owner. which allow an external party to verify a POR but do not provide any security guarantees when the user and/or external verifier are dishonest. OPOR provides users with the guarantee that their data is entirely stored in the cloud without having to verify their data themselves. Although auditors are made (legally) liable to monitor the availability of their files, users can verify the auditor’s work at any point in time; we show that this verification can be much less frequent, and is considerably more (computationally) efficient when compared to the verification of existing POR. We argue that OPOR is technically and economically viable. Indeed, by providing the necessary security guarantees for the auditors, OPOR enables auditors to issue a security SLA for the cloud users attesting that they will correctly verify the availability of outsourced data, in exchange, e.g., of financial remuneration. While the main barriers of wide adoption of the cloud lie in the lack of customer trust and in the high costs of deploying security measures in cloud infrastructures, OPOR bridges these gaps and enables customers and external auditors to establish a financial contract by which customers can rest assured that the security of their files is constantly monitored.

2.Literature Review:

Literature survey is the most important step in software development process. Before developing the tool it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, ten next steps are to determine which operating system and language can be used for developing the tool. Once the programmers start building the tool the programmers need lot of external support. This support can be obtained from senior programmers, from book or from websites. Before building the system the above consideration are taken into account for developing the proposed system.

3.System analysis and requirements Specification Existing System:

According to the role of the verifier in the model, all the schemes available fall into two categories:

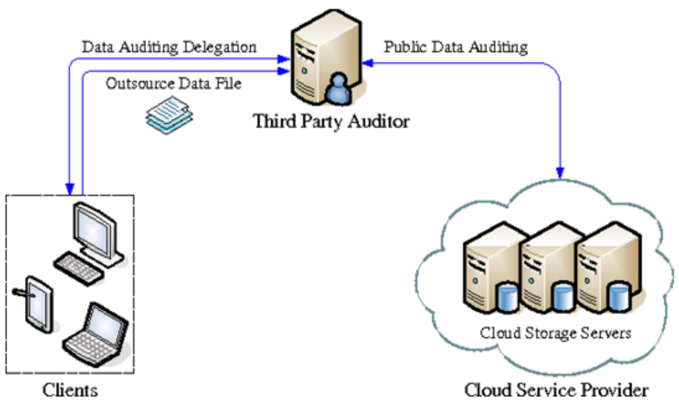
private verifiability and public verifiability. Although achieving higher efficiency, schemes with private verifiability impose computational burden on clients. On the other hand, public verifiability alleviates clients from performing a lot of computation for ensuring the integrity of data storage. To be specific, clients are able to delegate a third party to perform the verification without devotion of their computation resources. In the cloud, the clients may crash unexpectedly or cannot afford the overload of frequent integrity checks. Another major concern among previous designs that is the support of dynamic data operation for cloud data storage applications.

4.System Design:

Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.If the broader topic of product development "blends the perspective of marketing, design, and manufacturing into a single approach to product development," then design is the act of taking the marketing information and creating the design of the product to be manufactured. Systems design is therefore the process of defining and developing systems to satisfy specified requirements of the user.

Until the 1990s systems design had a crucial and respected role in the data processing industry. In the 1990s standardization of hardware and software resulted in the ability to build modular systems. The increasing importance of software running on generic platforms has enhanced the discipline of software engineering.Object oriented analysis and design methods are becoming the most widely used methods for computer systems design. The UML has become the standard language in object oriented analysis and design. It is widely used for modeling software systems and is increasingly used for high designing non-software systems and organizations.

4.1 System architecture:

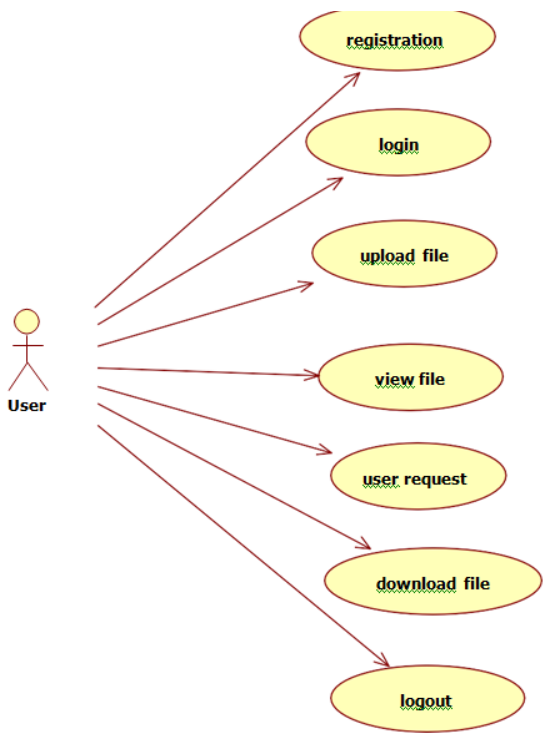


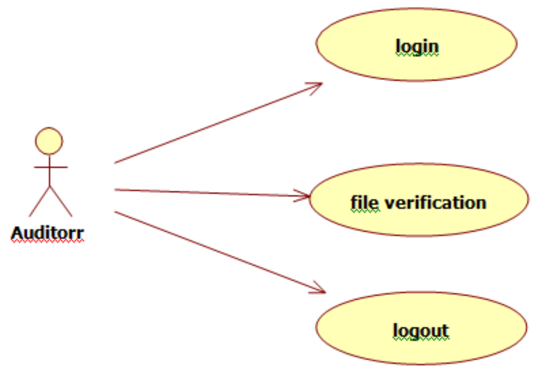
Use Case Diagram:

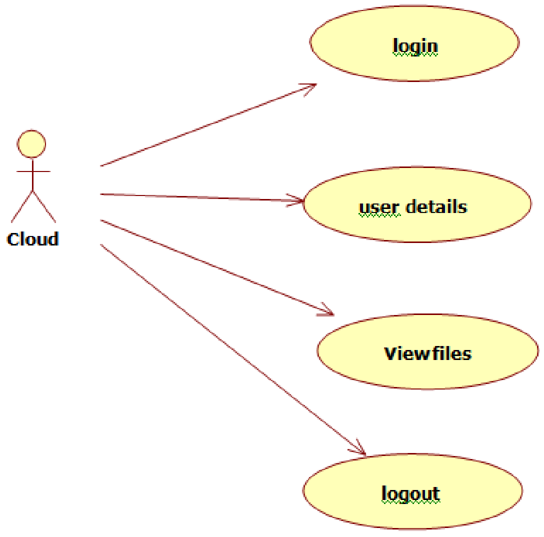
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

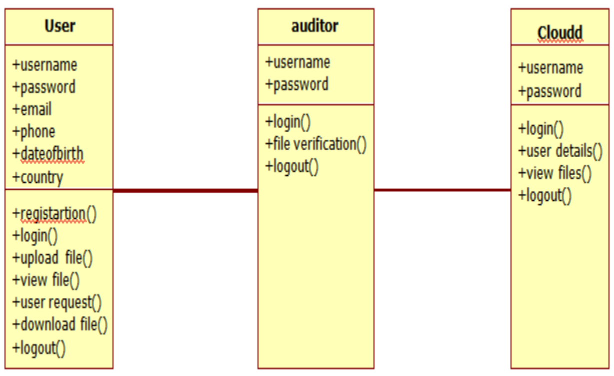






Class Diagram:

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal

6.System Implementation:

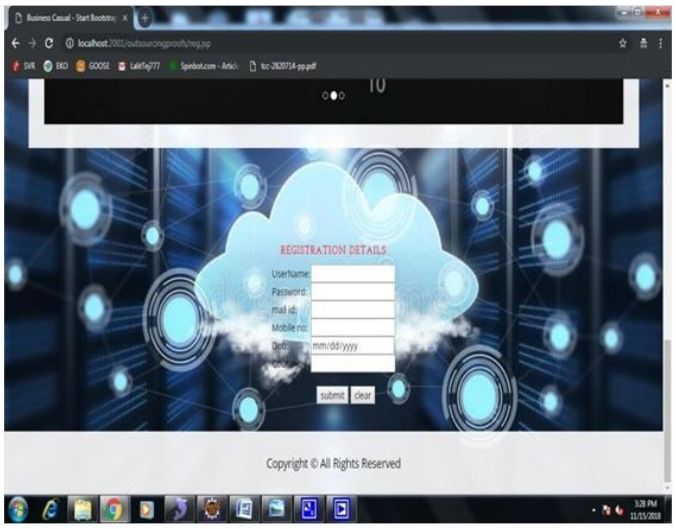
Many authorized users can access the remotely stored data from different geographic locations making it more convenient for them. Once the data has been outsourced to a remote CSP which may not be trustworthy, the data owners lose the direct control over their sensitive data. This lack of control raises new formidable and challenging tasks related to data confidentiality and integrity protection in cloud computing. The confidentiality issue can be handled by encrypting sensitive data before outsourcing to remote servers. As such, it is a crucial demand of customers to have strong evidence that the cloud servers still possess their data and it is not being tampered with or partially deleted over time. Consequently, many researchers have focused on the problem of provable data possession (PDP) and proposed different schemes to audit the data stored on remote servers. PDP is a technique for validating data integrity over remote servers. In a typical PDP model, the data owner generates some metadata/information for a data file to be used later for verification purposes through a *challenge-response* protocol with the remote/cloud server. The owner sends the file to be stored on a remote server which may be untrusted, and deletes the local copy of the file. As a proof that the server is still possessing the data file in its original form, it needs to correctly compute a response to a challenge vector sent from a verifier — who can be the original data owner or a trusted entity that shares some information with the owner.

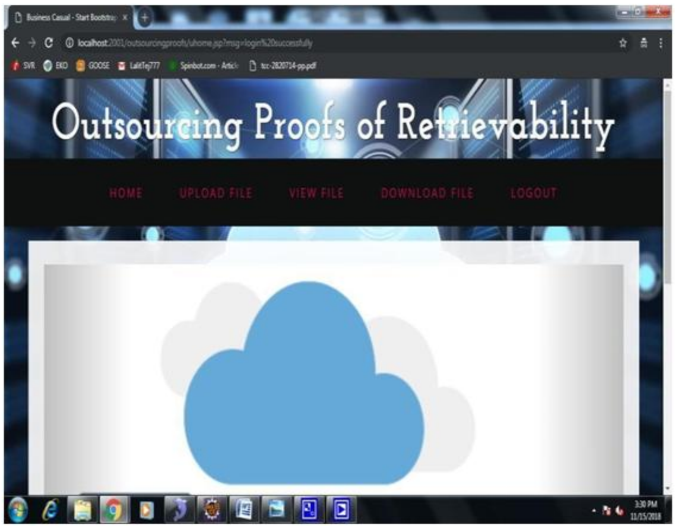
7.Test Cases:

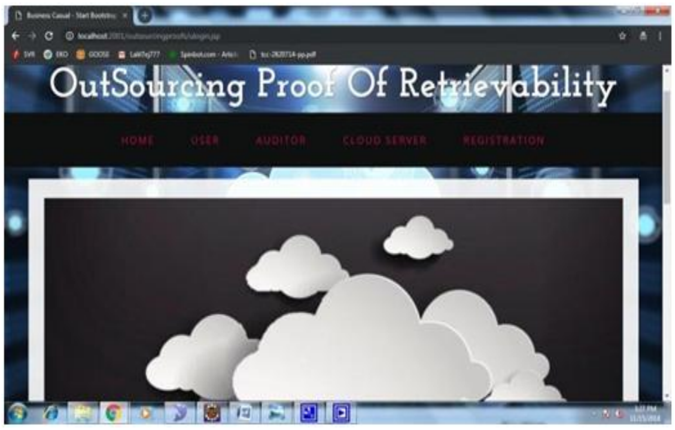
S. No	Test Case	Expected Output	Actual Output	Result
1	User	Enter correct username and password	Open User Page	Successful
2	User	User clicks on request sent for selected file	Decryption key will sent to mail by owner for downloading the selected file	Successful
3	Owner	Enter the correct username and password	Upload files and searches for user file requests & generates keys	Successful
4	Admin	Enter correct username and password	View the messages from owner & file action will be taken	Successful

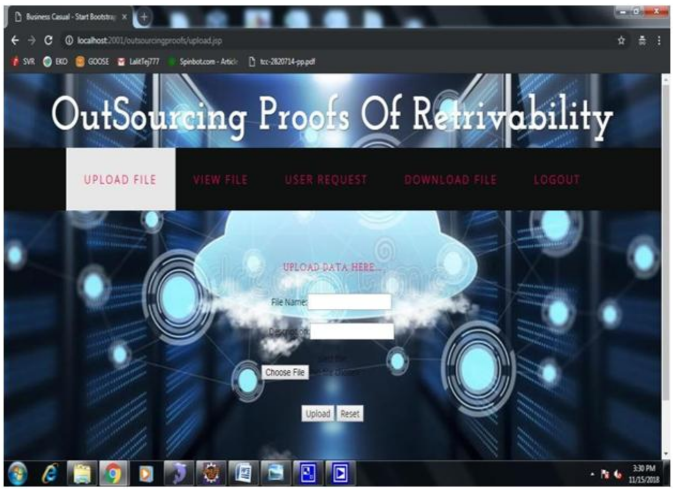
8.Results:

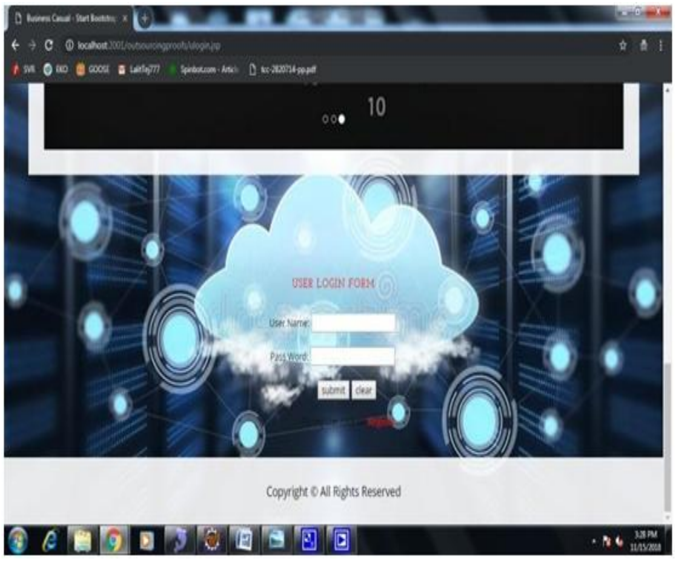
Screen Shots:

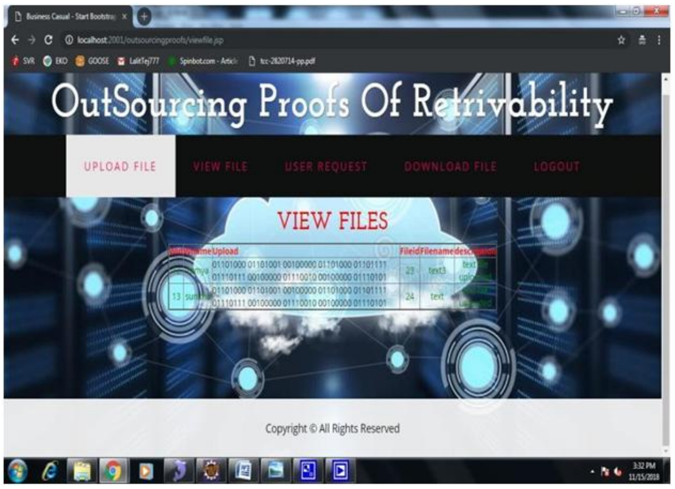

REGISTRATION DETAILS
UserName:
Password:
mail id:
Mobile no:
Dob:
mm/dd/yyyy
submit
clear
Copyright © All Rights Reserved


Outsourcing Proofs of Retrievability
HOME
UPLOAD FILE
VIEW FILE
DOWNLOAD FILE
LOGOUT

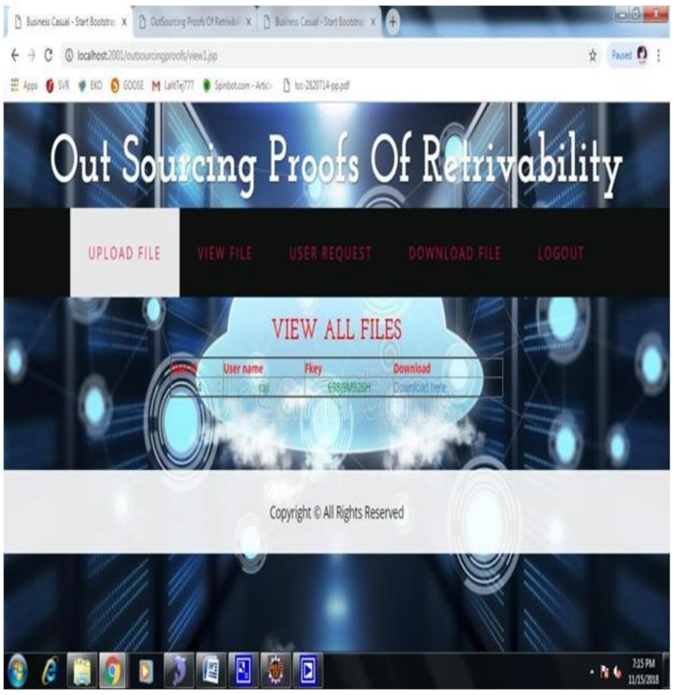

OutSourcing Proof Of Retrievability
HOME
USER
AUDITOR
CLOUD SERVER
REGISTRATION

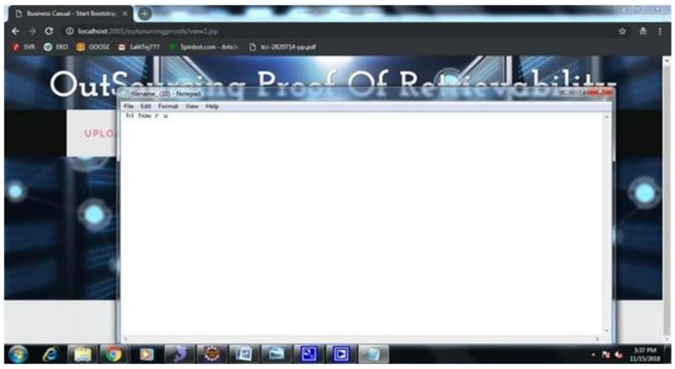

OutSourcing Proofs Of Retrivability
UPLOAD FILE
VIEW FILE
USER REQUEST
DOWNLOAD FILE
LOGOUT
UPLOAD DATA HERE...
File Name:
Choose File
Upload
Reset


USER LOGIN FORM
User Name:
Pass Word:
submit
clear
Copyright © All Rights Reserved

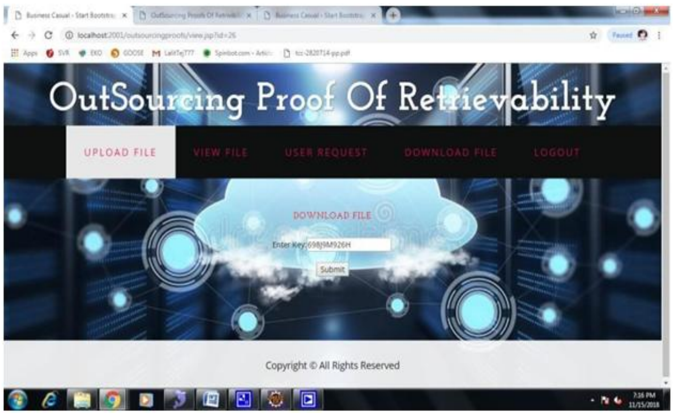

OutSourcing Proofs Of Retrivability
UPLOAD FILE
VIEW FILE
USER REQUEST
DOWNLOAD FILE
LOGOUT
VIEW FILES
Copyright © All Rights Reserved

ISSN No: 2348-4845
International Journal & Magazine of Engineering, Technology, Management and Research
A Peer Reviewed Open Access International Journal





Out Sourcing Proofs Of Retrivability
UPLOAD FILE
VIEW FILE
USER REQUEST
DOWNLOAD FILE
LOGOUT
VIEW ALL FILES
Copyright © All Rights Reserved



Conclusion:

In this paper, we introduced the notion of outsourced proofs of retrievability (OPOR), an extension of the traditional POR concept, and proposed an efficient instantiation of OPOR, dubbed Fortress. We implemented a prototype based on Fortress, and evaluated its performance in a realistic cloud setting. Our results show that our proposal incurs minimal overhead on the user and scales well with the number of users. We argue that Fortress motivates a novel business model in which customers and external auditors establish a contract by which customers can rest assured about the security of their files. By doing so, Fortress increases the users' trust in the cloud, while incurring minimal user interaction. We therefore argue that our work lays basic foundations for realizing secure external auditing of cloud services; we believe that such auditor-based schemes will provide a stepping stone for establishing a cyber-insurance market for cloud services. In terms of future work, we plan to explore efficient mechanisms to optimize the store procedure in Fortress, to investigate a generic transformation to turn any POR into an OPOR, and to design an OPOR scheme that supports dynamic updates of the stored file.

References:

[1].Bitcoin real-time stats and tools.http://blockexplorer.com/q.

[2].CloudComputing:CloudsecurityConcernshttp://technet.microsoft.com/enus/magazine/hh536219.aspx.

[3]PBCLibrary.http://crypto.stanford.edu/pbc/, 2007.

[4]Jerasure.https://github.com/tsuraan/Jerasure, 2008.

[5]Amazon S3 Service Level Agreement, 2009. http://aws.amazon.com/ s3-sla/.

[6]MicorosfotCorporation. Windows Azure Pricing and Service Agreement, 2009.

[7]JPBC:Java Pairing-Based Cryptography Library. http://gas.dia.unisa.it/ projects/jpbc/#.U3HBFfna5cY, 2013.

[8]Protect data stored and shared in public cloud storage.http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/ Documents/Dell_Data_Protection_Cloud_Edition_Data_Sheet.pdf, 2013.

[9]Bit coin as a public source of randomness. https://docs.google.com/presentation/d/1VWHm4Moza2znhXSOJ8FacfNK2B_vxnfbdZgC5EpeXFE/view?pli=1#slide=id.g3934beb89_034, 2014.

[10]Google loses data after lightning strikes. http://money.cnn.com/2015/08/ 19/technology/google-data-loss-lightning/, 2015.

[11]SatoshiNakamoto. Bitcoin: A Peer-to-Peer Electronic CashSystem.

[12]Frederik Armknecht, Jens-Matthias Bohli, Ghassan O. Karame, Zongren Liu, andChristian A. Reuter. Outsourced proofs of retrievability. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, Scottsdale, AZ, USA,November 3-7, 2014*, pages 831–843, 2014

[13]Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary N. J. Peterson, and Dawn Xiaodong Song. Provable data possession at untrusted stores. In *ACM Conference on Computer and Communications Security*, pages 598–609, 2007.

[14]Giuseppe Ateniese, Roberto Di Pietro, Luigi V. Mancini, and Gene Tsudik. Scalable and efficient provable data possession. *IACR Cryptology ePrint Archive*, 2008:114, 2008.