

## KEY GENERATION BASED PROXY SERVER USING PROXY KEY PROTOCOL

<sup>1</sup>Neeraja Menon, <sup>2</sup>N.Ashok Kumar,

<sup>1</sup>PG Scholar, Department of Computer Science and Engineering , Maharaja Engineering College,  
Avinashi.

<sup>2</sup>Assistant Professor, Department of Computer Science and Engineering , Maharaja Engineering College,  
Avinashi.

### ABSTRACT

The main objective of this project is to develop a Computational Private Information Retrieval protocols on cloud architecture using erasure code for secured data forwarding. These protocols are too costly in practice because they invoke complex arithmetic operations for every bit of the database. Basically cloud storage architecture will have a collection of storage servers with higher end configuration which will provides long-term storage services over the Internet and also for the cloud storage system. Here storing and retrieving the data in a third party's cloud system and public auditing scheme causes serious problems and conflict over data confidentiality during the data transactions. Whenever third party big data storage will involved with the cloud server this conflict will occur naturally. Even thou there are various methods are available to overcome this problem like cryptography, key encryption and etc. But general encryption schemes protect data confidentiality during the transaction, but along with this process the main drawback will, it limits the functionality of the storage system. This is because; a few operations only supported over encrypted data. These methods will cause failure. In order to constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority.

**Keywords:** cryptography, secure storage system, cloud storage architecture.

### 1. INTRODUCTION

A cloud storage system, consisting of a collection of storage servers, provides long-term storage services over the Internet. Storing data in a third party's cloud system causes serious concern over data confidentiality. General encryption schemes protect data confidentiality, but also limit the functionality of the storage system because a few operations are supported over encrypted data. Using Cloud Storage, users can remotely store their data and enjoy the on-demand high quality applications and services from a shared pool of configurable computing resources, without the burden of local data storage and maintenance. However, the fact that users no longer have physical possession of the outsourced data makes the data integrity protection in Cloud Computing a formidable task, especially for users with constrained computing resources. Moreover, users should be able to just use the cloud storage as if it is local, without worrying about the need to verify its integrity. Thus, enabling public audit ability for cloud storage is of critical importance so that users can resort to a third party auditor (TPA) to check the

integrity of outsourced data and be worry-free. To securely introduce an effective TPA, the auditing process should bring in no new vulnerabilities towards user data privacy, and introduce no additional online burden to user. In this paper, we propose a secure cloud storage system supporting privacy-preserving public auditing. We further extend our result to enable the TPA to perform audits for multiple users simultaneously and efficiently. Extensive security and performance analysis show the proposed schemes are provably secure and highly efficient. . The storage server will be unique which has been distributed into much system for easy access of data. It contains only the encrypted data of the data owners.

## 2. LITERATURE SURVEY

We consider the problem of constructing an erasure code for storage over a network when the data sources are distributed. Specifically, we assume that there are  $n$  storage nodes with limited memory and  $k < n$  sources generating the data. We want a data collector, who can appear anywhere in the network, to query any  $k$  storage nodes and be able to retrieve the data. We introduce Decentralized Erasure Codes, which are linear codes with a specific randomized structure inspired by network coding on random bipartite graphs. We show that decentralized erasure codes are optimally sparse, and lead to reduced communication, storage and computation cost over random linear coding. In this correspondence, we address the problem of distributed networked storage when there are multiple, distributed sources that generate data that must be stored efficiently in multiple storage nodes, each having limited memory.

Plautus is a cryptographic storage system that enables secure file sharing without placing much trust on the file servers. In particular, it makes novel use of cryptographic primitives to protect and share files. Plautus features highly scalable key management while allowing individual users to retain direct control over who gets access to their files. We explain the mechanisms in Plautus to reduce the number of cryptographic keys exchanged between users by using file groups, distinguish file read and write access, handle user revocation efficiently, and allow an untrusted server to authorize file writes. We have built a prototype of Plautus on Open AFS. Measurements of this prototype show that Plautus achieves strong security with overhead comparable to systems that encrypt all network traffic. As storage systems and individual storage devices themselves become networked, they must defend both against the usual attacks on messages traversing an untrusted, potentially public, network as well as attacks on the stored data itself. This is a challenge because the primary purpose of networked storage is to enable easy sharing of data, which is often at odds with data security. *Networked Systems Design and Implementation (NSDI)*, pp. 337-350, 2010.

Availability is a storage system property that is both highly desired and yet minimally engineered. While many systems provide mechanisms to improve availability – such as redundancy and failure recovery – how to best configure these mechanisms is typically left to the system manager. Unfortunately, few individuals have the skills to properly manage the trade-offs involved, let alone the time to adapt these decisions to changing conditions. Instead, most systems are configured statically and with only a cursory understanding of how the configuration will impact overall performance or availability.

### 3. MODULE DESCRIPTION

The public cloud environment is the IaaS/PaaS Infrastructure or Platform as a Service that we rent from Linux (IaaS) or Microsoft (PaaS). Both are enabled for web hosting. Then, your SaaS stack will run under your Internet environment most likely in a virtualized one on your own equipment which would make it private. In this project we specialize in private cloud technology. Here we execute in a cloud environment. If strict security requirements go public or hybrid and if not, try the public or community cloud environment. So that here we are implementing a web services for the output purpose as well as the environment will be shown in actual while hosting the application. So finally SaaS can be fully utilized in cloud environment as IaaS/PaaS. We consider the problem of constructing an erasure code for storage over a network when the data sources are distributed in the cloud server. Specifically, we assume that there are  $n$  storage nodes with limited memory and  $k < n$  sources generating the data. We want a data collector, who can appear anywhere in the network for accessing the data, to query any  $k$  storage nodes and be able to retrieve the data. We introduce Decentralized Erasure Codes, which are linear codes with a specific randomized structure inspired by network coding on random bipartite graphs with encrypted format. We show that decentralized erasure codes are optimally sparse, and lead to reduced communication, storage and computation cost over random linear coding over the cloud server. It is one of the advanced encryption model which works on both real system and virtual systems. This works more efficient on cloud systems. Proxy re-encryption schemes are cryptosystems which allow third-parties (proxies) to alter a cipher text which has been encrypted for one party, so that it may be decrypted by another. Proxy re-encryption schemes are similar to traditional symmetric or asymmetric encryption schemes. It allows a message recipient (key holder) to generate a re-encryption key based on his secret key and the key of the delegated user. This re-encryption key is used by the proxy as input to the re-encryption function, which is executed by the proxy to translate cipher texts to the delegated user's key. Asymmetric proxy re-encryption schemes come in bi-directional and uni directional varieties. Proxy re-encryption schemes allow for a cipher text to be re-encrypted an unlimited number of times. Proxy re-encryption should not be confused with proxy signatures, which is a separate construction with a different purpose.

### 4. PROTOCOL IMPLEMENTATION - Method

According to PKP, the reference implementation is as follows:

- Generate a key,  $K$ , which is an arbitrary by testring, and share it securely with the client.
- Agree upon an epoch,  $T_0$ , and an interval,  $T_I$ , which will be used to calculate the value of the counter  $C$  (defaults are the Unix epoch as  $T_0$  and 30 seconds as  $T_I$ )
- Agree upon a cryptographic hash method (default is SHA-1)
- Agree upon a token length,  $N$  (default is 6)

Although RFC 6238 allows different parameters to be used, the Google implementation of the authenticator app does not support  $T_0$ ,  $T_I$  values, hash methods and token lengths different from the

default. It also expects the K secret key to be entered (or supplied in a QR code) in base-32 encoding according to RFC 3548.

Once the parameters are agreed upon, token generation is as follows:

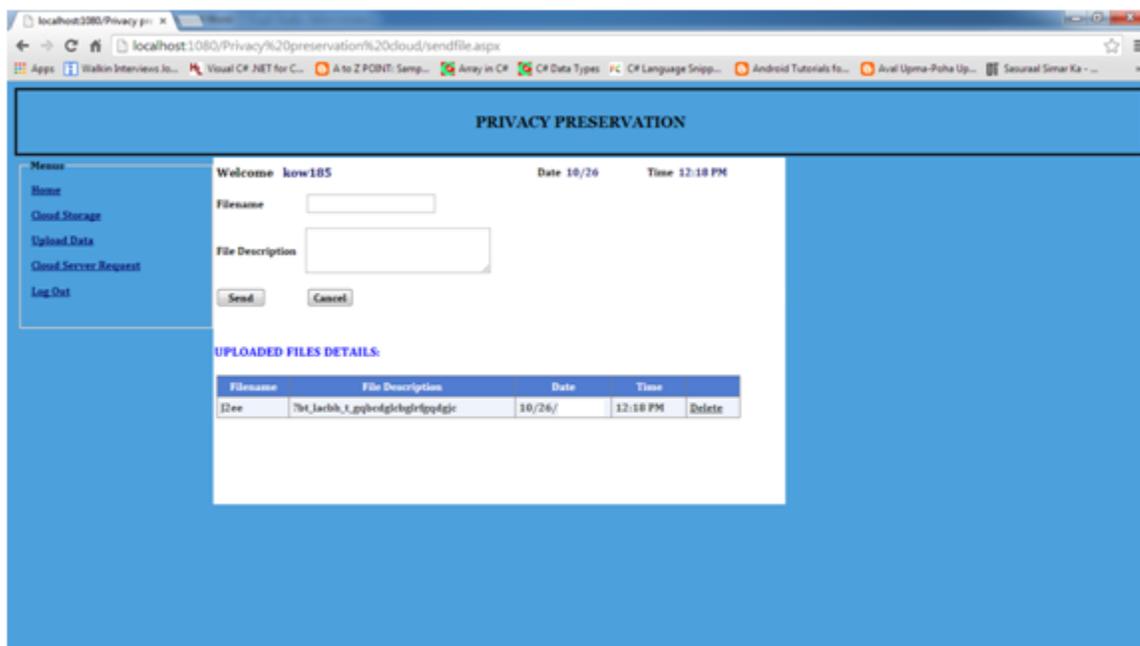
1. Calculate C as the number of times TI has elapsed after T0.
2. Compute the HMAC hash H with C as the message and K as the key (the HMAC algorithm is defined in the previous section, but also most cryptographically libraries support it). K should be passed as it is, C should be passed as a raw 64-bit unsigned integer.
3. Take the least 4 significant bits of H and use it as an offset, O.
4. Take 4 bytes from H starting at O bytes MSB, discard the most significant bit and store the rest as an (unsigned) 32-bit integer, I.
5. The token is the lowest N digits of I in base 10. If the result has fewer digits than N, pad it with zeroes from the left.

Both the server and the client compute the token, then the server checks if the token supplied by the client matches the locally generated token. Some servers allow codes that should have been generated before or after the current time in order to account for slight clock skews, network latencies and user delays

## 5. RESULT AND DISCUSSION (IMPLEMENTATION)

The Result obtained according to the committed abstract.

This screen shows that the user uploaded data in the cloud server in the encrypted format. This will be done through the re encryption method. This can be done through the prior user name and password.



Encryption type = 64 bit key

Key Length = 23 char

Key type = Alpha numerical with special characters.

Key character = Encryption and decryption

Key Limitation = Caps alphabets = 26, Small alphabets = 26, 0 -9 numbers = 10, Special Characters = 10:  
result = **3.848329407410064e+135** of key combinations can be produced.

It is equivalent to 30000 trillion and above combination

## CONCLUSION

Thus we are concluding that all the result obtained according to the committed abstract. In this paper, we consider a cloud storage system consists of storage servers and key servers. We integrate a newly proposed threshold proxy re-encryption scheme and erasure codes over exponents. The threshold proxy re-encryption scheme supports encoding, forwarding, and partial decryption operations in a distributed way. To decrypt a message of  $k$  blocks that are encrypted and encoded ton code word symbols, each key server only has to partially decrypt two codeword symbols in our system. By using the threshold proxy re-encryption scheme, we present a secure cloud storage system that provides secure data storage and secure data forwarding functionality in a decentralized structure. Moreover, each storage server independently performs encoding and re-encryption and each key server independently perform partial decryption. Our storage system and some newly proposed content addressable file systems and storage systems are highly compatible. Our storage servers act as storage nodes in a content addressable storage system for storing content addressable blocks. Our key servers act as access nodes for providing a front-end layer such as a traditional file system interface. Further study on detailed cooperation is required.

## REFERENCES

- [1] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G.M. Voelker, "Total Recall: System Support for Automated Availability Management," Proc. First Symp. Networked Systems Design and Implementation (NSDI), pp. 337-350, 2004.
- [2] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Ubiquitous Access to Distributed Data in Large-Scale Sensor Networks through Decentralized Erasure Codes," Proc. Fourth Int'l Symp. Information Processing in Sensor Networks (IPSN), pp. 111-117, 2005.
- [3] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, "Decentralized Erasure Codes for Distributed Networked Storage," IEEE Trans. Information Theory, vol. 52, no. 6 pp. 2809-2816, June 2006.
- [4] M. Mambo and E. Okamoto, "Proxy Cryptosystems: Delegation of the Power to Decrypt Ciphertexts," IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences, vol. E80-A, no. 1, pp. 54-63, 1997.

- [5] M. Blaze, G. Bleumer, and M. Strauss, "Divertible Protocols and Atomic Proxy Cryptography," Proc. Int'l Conf. Theory and Application of Cryptographic Techniques (EUROCRYPT), pp. 127-144, 1998.
- [6] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, 2006.
- [7] Q. Tang, "Type-Based Proxy Re-Encryption and Its Construction," Proc. Ninth Int'l Conf. Cryptology in India: Progress in Cryptology (INDOCRYPT), pp. 130-144, 2008.
- [8] G. Ateniese, K. Benson, and S. Hohenberger, "Key-Private Proxy Re-Encryption," Proc. Topics in Cryptology (CT-RSA), pp. 279-294, 2009.
- [9] J. Shao and Z. Cao, "CCA-Secure Proxy Re-Encryption without Pairings," Proc. 12th Int'l Conf. Practice and Theory in Public Key Cryptography (PKC), pp. 357-376, 2009.
- [10] 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), pp. 598-609, 2007.
- [11] 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), pp. 1-10, 2008