RESEARCH ARTICLE OPEN ACCESS

Securely Data Forwarding and Maintaining Reliability of Data in Cloud Computing

Sonali A.Wanjari*, Asst. Prof. Bharat Tidke**
*(Research Scholar, Department of Computer Network, Flora institute technology (FIT) Khopi, Pune-412205 Maharashtra, India)
** (Assistant Professor, Department of Computer Network, Flora institute technology (FIT) Khopi, Pune-412205 Maharashtra, India)

ABSTRACT
Cloud works as an online storage servers and provides long term storage services over the internet. It is like a third party in whom we can store a data so they need data confidentiality, robustness and functionality. Encryption and encoding methods are used to solve such problems. After that divide proxy re-encryption scheme and integrating it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure, robust data storage and retrieval but also lets the user forward his data to another user without retrieving the data. A concept of backup in same server allows users to retrieve failure data successfully in the storage server and also forward to another user without retrieving the data back. This is an attempt to provide light-weight approach which protects data access in distributed storage servers. User can implement all important concept i.e. Confidentiality for security, Robustness for healthy data, Reliability for flexible data, Availability for compulsory data will be achieved to another user which is store in cloud and easily overcome problem of "Securely data forwarding and maintaining, reliability of data in cloud computing "using different type of Methodology and Technology.

Keywords **-** Homomorphic Encryption method, encoding method, proxy re-encryption scheme, decentralized erasure code, cloud computing.

I. INTRODUCTION

With the advent of faster and better network options and universal internet access many services are available online anytime and anywhere access. Cloud is a technology in which data is stored in different servers. Cloud computing concepts are used for managing data on a cloud. End Users may not be aware of computing options. However, these programs work without the knowledge of a user and its controls and manage most of the data tasks.

Cloud computing is surrounded by many security issues like securing data, examining the utilization of cloud by the different cloud computing vendors. Cloud computing is used to store a data in cloud storage system so it provide robustness, privacy and reliability.

Cloud storage system is a large-scale distributed storage system that consists of many independent storage servers. Data robustness is a major requirement for storage systems. Each user usually gives a secret key which is generated by him. The user can store, forward and retrieve data in the cloud only after using secret key. If the user loses his key, he is blocked from the system. This system is used to avoid unauthorized access to cloud database. The authorized user may give a second chance to retrieve his data. Therefore, when the user loses his secret key twice then only he will be blocked from the system. Because of the huge amount of data stored by a cloud efficient processing and analysis of data has become a big challenge.

Data robustness is a major requirement for storage systems. There have been many proposals of storing data over storage servers [1], [2], [3], [4]. One way to provide data robustness is to replicate a message such that each storage server stores a copy of the message. It is very robust because the message can be retrieved as long as one storage server survives. Another way is to encode a message of k symbols into a code word of n symbols by erasure coding. To store a message, each of its code word symbols is stored in a different storage server. A decentralized erasure code is an erasure code that independently computes each code word symbol for a message. Thus, the encoding process for a message can be split into n parallel tasks of generating code word symbols. A decentralized erasure code is suitable for use in a distributed storage system. After the message symbols are sent to storage servers, each storage server independently computes a code word symbol for the received message symbols and stores it. This finishes the encoding and storing process. Storing data in a Third party's cloud system causes serious concern on data confidentiality. In order to provide strong confidentiality for messages in storage servers, a user can encrypt messages by a

cryptographic method before applying an erasure code method to encode and store messages. When he wants to use a message, he needs to retrieve the code word symbols from storage servers, decode them, and then decrypt them by using cryptographic keys. To well fit the distributed structure of systems; user requires that servers independently perform all operations. With this consideration, user proposes a new threshold proxy re-encryption scheme and integrates it with a secure decentralized code to form a secure distributed storage system. The encryption scheme supports encoding operations over encrypted messages and forwarding operations over encrypted and encoded messages. The tight integration of encoding, encryption, and forwarding makes the storage system efficiently meet the requirements of data robustness, data confidentiality, and data forwarding. Accomplishing the integration with consideration of a distributed structure is challenging. Our system meets the requirements that storage servers independently perform encoding and re-encryption and key servers independently perform partial decryption [5].

II. LITERATURE SURVEY

The In [1] J. Kubiatowicz, et.al author produced an urgent need for Persistent information. In this paper presented Ocean Store, a Utility infrastructure designed to span the globe and provide secure, highly available access to persistent objects. Several properties distinguish Ocean Store from other systems: the utility model, the untrusted infrastructure, support for truly nomadic data, and use of introspection to enhance performance and maintainability. A utility model makes the notion of a global system possible, but introduces the possibility of untrustworthy servers in the system. To this end, assume that servers may be run by Adversaries and cannot be trusted with clear text; as a result, server side recently on encrypted information. Nomadic data permits a wide range of optimizations for access to information by bringing it "close" to where it is needed, and enables rapid response to regional outages and denial-of-service attacks. These optimizations are assisted by introspection, the continuous online collection and analysis of access patterns. Ocean Store is under construction. The Ocean Store system has two design goals that differentiate it from similar systems: The ability to be constructed from an untrusted infrastructure and Support of nomadic data.

In [2] P. Druschel and A. Rowstron, author describes PAST, a large-scale, Internet based, global storage utility that provides high availability, persistence and protects the anonymity of clients and storage providers. PAST is a peer-to-peer Internet application and is entirely self-organizing. PAST nodes serve as access points for clients, participate in the routing of client requests, and contribute storage to the system. Nodes are not trusted; they may join the system at any time and may silently leave the system without warning. Yet, the system is able to provide strong assurances, efficient storage access, load balancing and scalability. In [3] Adya, W.J. Bolosky, M. Caistro, et.al author Far site is a scalable, decentralized, network file system where in a loosely coupled collection of insecure and unreliable machines collaboratively establishes a virtual file server that is secure and reliable. Far site provides the shared namespace, location-transparent access, and reliable data storage of a central file server and also the low cost, decentralized security, and privacy of desktop workstations. It requires no central-administrative effort apart from signing user and machine certificates. Far site's core architecture is a collection of interacting, Byzantine-fault-tolerant replica groups, arranged in a tree that overlays the file-system namespace hierarchy. Because the vast majority of file-system data is opaque file content, Far site maintains only indirection pointers and cryptographic checksums of this data as part of the Byzantine-replicated state. Actual content is encrypted and stored using raw (non-Byzantine) replication; however, the architecture could alternatively employ erasure-coded replication to improve storage efficiency.

In [4] A. Haeberlen, A. Mislove, and P. Druschel, author Decentralized storage systems aggregate the available disk space of participating computers to provide a large storage facility. These systems rely on data redundancy to ensure durable storage despite of node failures. However, existing systems either assume independent node failures, or they rely on introspection to carefully place redundant data on nodes with low expected failure correlation. Unfortunately, node failures are not independent in practice and constructing an accurate failure model is difficult in large-scale systems. At the same time, malicious worms that propagate through the Internet pose a real threat of large-scale correlated failures. Such rare but potentially catastrophic failures must be considered when attempting to provide highly durable storage. In this paper, Glacier describe, a distributed storage system that relies on massive redundancy to mask the effect of large-scale correlated failures. Glacier is designed to aggressively minimize the cost of this redundancy in space and time: Erasure coding and garbage collection reduces the storage cost; aggregation of small objects and a loosely coupled maintenance protocol for redundant fragments minimizes the messaging cost. In one configuration, for instance, our system can provide six-nines durable storage despite correlated failures of up to 60% of the storage nodes, at the cost of an eleven-fold storage

overhead and an average messaging overhead of only 4 messages per node and minute during normal operation.

In [5] H.Y. Lin and W.G. Tzeng, author consider a cloud storage system consists of storage servers and key servers. User integrates 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 to n code word symbols, each key server only has to partially decrypt two code word symbols in our system. By using the threshold proxy re-encryption scheme, 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 system are highly compatible.

In [6] D.R. Brown bridge, L.F. Marshall, and B. Rendell, author paper a software subsystem that can be added to each of a set of physically interconnected UNIX or UNIX look-alike systems, so as to construct a distributed system which is functionally indistinguishable at both the user and the program level from a conventional single processor UNIX system. The techniques used are applicable to a variety and multiplicity of both local and wide area networks, and enable all issues of inter-processor communication, network protocols, etc., to be hidden. A brief account is given of experience with such a distributed system, which is currently operational on a set of PDP11s connected by a Cambridge Ring.

In [7] A.G. Demakis, et.al author describe A large-scale wireless sensor network of n nodes, where a fraction k out of n generates data packets of global interest. Assuming that the individual nodes have limited storage and computational capabilities, the problem is how to enable ubiquitous access to the distributed data packets. Specifically, each node can store at most one data packet, and study the problem of diffusing the data so that by querying any k nodes, it is possible to retrieve all the k data packets of interest (with high probability).user introduce a class of erasure codes and show how to solve this problem efficiently in a completely distributed and robust way. Specifically user can efficiently diffuse the data by .pre-routing only O (lnn) packets per data node to randomly selected storage nodes. By using the proposed scheme, the distributed data becomes available .at the fingertips of a potential data collector located anywhere in the network.

In [8] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, author create 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. A motivating application, one can think of sensor networks where the sensor measurements are inherently distributed and sensor motes have constrained communication, computation, and storage capabilities. In addition, distributed networked storage can be useful for peer-to-peer networks or redundant arrays of independent disks (RAID) systems. The distributed sources are k data nodes, each producing one data packet of interest. Also assume n storage nodes that will be used as a distributed network memory. If each storage node can store one data packet, to diffuse the data packets so that by querying any k storage nodes, it is possible to retrieve all the k data packets of interest (with high probability). The key issue, of course, is whether it is possible to achieve this robust distributed storage with minimal computation and communication. To solve this problem, proposed decentralized erasure codes, which are randomized linear codes with a specific probabilistic structure that leads to optimally sparse generator matrices. These codes can be created by a randomized network protocol where each data node “pre-routes” its data packet O (log n) randomly and independently selected storage nodes each storage node creates a random linear combination of whatever it happens to receive. Therefore each node operates autonomously without any central points of control and with small communication cost. The problem of constructing an erasure code for storage over a network when the data sources are distributed. Specifically, assume that there are n storage nodes with limited memory and $k < n$ sources generating the data. User want a data collector, who can appear anywhere in the network, to query any k storage nodes and be able to retrieve the data.

In [9] G. Ateniese, K. Fu, et.al author allows a proxy to transform a cipher text computed under Alice’s public key into one that can be opened by Bob’s secret key. There are many useful applications of this primitive. For instance, Alice might wish to temporarily forward encrypted email to her colleague Bob, without giving him her secret key. In this case, Alice the delegator could designate a proxy to re-encrypt her incoming mail into a format that Bob the delegate can decrypt using his own secret key. Alice could simply provide her secret key to the proxy, but this requires an unrealistic level of trust in the proxy. User presents several efficient proxy re-encryption schemes that offer security improvements over earlier approaches. The primary advantage of our schemes is that they are and do not

require delegators to reveal their entire secret key to anyone – or even interact with the delegate – in order to allow a proxy to re-encrypt their cipher texts. In our schemes, only a limited amount of trust is placed in the proxy. This enables a number of applications that would not be practical if the proxy needed to be fully trusted. User provides the first empirical performance measurements of applications using proxy re-encryption.

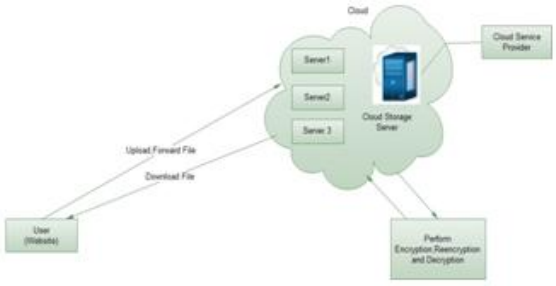
In [10] Q. Tang, author recently, the concept of proxy re-encryption has been shown very useful in a number of applications, especially in enforcing access control policies. In existing proxy re-encryption schemes, the delegate can decrypt all cipher texts for the delegator after re-encryption by the proxy. Consequently, in order to implement fine-grained access control policies, the delegator needs to either use multiple key pairs or trust the proxy to behave honestly. In this paper, user extends this concept and proposes type-based proxy re-encryption, which enables the delegator to selectively delegate his decryption right to the delegate while only needs one key pair. As a result, type-based proxy re-encryption enables the delegator to implement fine-grained policies with one key pair without any additional trust on the proxy. User provides a security model for our concept and provides formal definitions for semantic security and cipher text privacy which is a valuable attribute in privacy-sensitive contexts. User proposes two type-based proxy re-encryption schemes: one is CPA secure with cipher text privacy while the other is CCA secure without cipher text privacy.

In [11] G. Ateniese, K. Benson, and S. Hohenberger, author allows a proxy to convert a cipher text encrypted under one key into an encryption of the same message under another key. The main idea is to place as little trust and reveal as little information to the proxy as necessary to allow it to perform its translations. At the very least, the proxy should not be able to learn the keys of the participants or the content of the messages it re-encrypts. However, in all prior PRE schemes, it is easy for the proxy to determine between which participants a re-encryption key can transform cipher texts. In this work, users propose key private re-encryption keys as an additional useful property of PRE schemes. A definition of what it means for a PRE scheme to be secure and key-private.

In [12] C. Wang, Q. Wang, et.al author show how to divide data D into n pieces in such a way that D is easily reconstruct able from any k pieces, but even complete knowledge of k - 1 piece reveals absolutely no information about D. This technique enables the construction of robust key management schemes for cryptographic systems that can function securely and reliably even when misfortunes destroy half the pieces and security breaches expose all but one of the remaining pieces.

III. SYSTEM MODEL

In Figure 3 shows the Existing Architecture System.



Here describe the decentralized erasure code which is used to split up the messages or text data into n number of blocks. This is used for splitting purpose. In this paper the results n=AKC allows that number of storage server be greater than the number of blocks of a text data's. The encoding method for a message can be split into n parallel tasks of generating codeword symbols. A decentralized erasure code is used in a distributed storage system. The n blocked message is stored in for the integration process.

In an integration processes, the divided message is joined into an m number of blocks, and stored into larger storage server. User A encrypts his message M is decomposed **System Model** into k number of blocks m1, m2, …. Mk and which has an identifier ID. User A encrypts each block mi into a cipher text Ci and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a code word symbol and stores it. Integration is used to combine messages into m number of block, which is encrypted and stored into a large number storage server. Then forward to user B. Data which is encrypted by using single key. This is produced by using hash key algorithm. In the data storage phase, user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k number of blocks m1, m2.., Mk and which has an identifier ID. User A encrypts each block mi into a cipher text Ci and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a code word symbol and stores it.

IV. Proposed Work

In Objective of this model is achieve the reliability of the software and increases role of distributed storage servers. Objectives are to manage and improve:

1. Data should be encrypted before stored.
2. Data divide into multiple parts before storing
3. Data should be re-encrypt while forwarding
4. After re-encrypt secrete key must be generated
5. Create back up of stored Data in same distributed servers

SET THEORY ANALYSIS:

Let 'S' be the communication system S= {L, W}; L – LAN Connection- Wi-Fi Connection, Identify the inputs I= {A, F, SK, PK, T} Where-A=Authority= File that user wants to upload on the Storage; SK= Secrete Key of New User; PK= Public Key; T = Tokens.

Identify the Outputs: Let O be the se of outputs O = {D, P, F} Where-D= Download File; P=Private Key; F=Forward File.

Identify set of Function: Let F be the set of Functions-F= {F1, F2, F3, F4} Where-F1 – Upload File; F2 – Encrypt File;F3- Generate Public and Private Key;F4- Download Own File;F5- Forward File; F6- Get Secrete Key; F7- Download Forwarded File.

Final State-Fs =All communication is done. Failure case-FL=Network failure. Constraints -Φ be the constraints Φ= {N} Where N= Number of Servers

V. Algorithm

1. Key Generation phase using RSA (Rivest Shamir Adelman) Algorithm

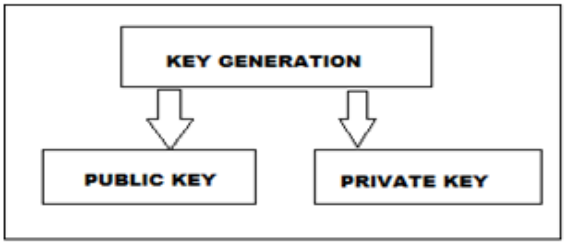


FIGURE 1:KEY GENERATION PHASE

In Figure.1 shows the Key Generation Phase. Here describes the Algorithm two key requires one is public key and another is Private Key. While encryption public key is required and during Decryption public (PKA) and secret key (SKA) both are required.

2. Data Storage Algorithm

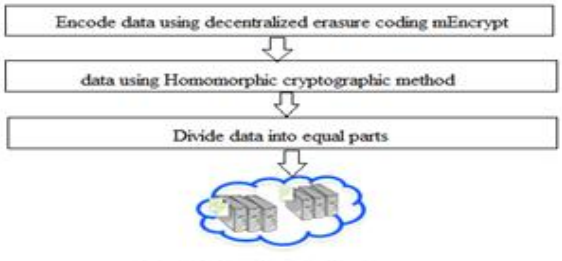


Figure 2: Flowchart for Data Storage

In Figure 2 shows the Flowchart for Data Storage. Here describes the algorithm user A encrypts his message M and dispatches it to storage servers. A message M is decomposed into k blocks m1; m2; . . .; Mk and has an identifier ID. User A encrypts each block mi into a cipher text Ci and sends it to v randomly chosen storage servers. Upon receiving cipher texts from a user, each storage server linearly combines them with randomly chosen coefficients into a code word symbol and stores it.

3. Data forwarding Algorithm

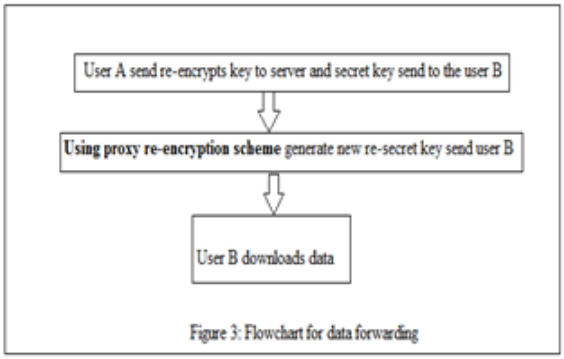


Figure 3: Flowchart for data forwarding

The data forwarding phase shown in Figure 3, user A forwards his encrypted message with an identifier ID stored in storage servers to user B such that B can decrypt the forwarded message by his secret key. To do so, a uses his secret key SKA and B's public key PKB to compute a re-encryption key RKID A! B and then sends RKID A! B to all storage servers. Each storage server uses the re encryption key to re-encrypt its code word symbol for later retrieval requests by B. The re-encrypted code word symbol is the combination of cipher texts under B's public key. In order to distinguish re-encrypted code word symbols from intact ones, call them original code word symbols and re encrypted code word symbols, respectively.

4. Data Retrieve Algorithm

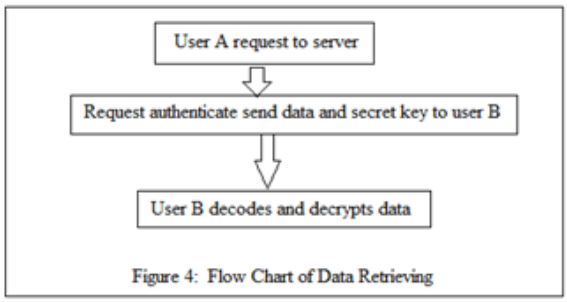


Figure 4: Flow Chart of Data Retrieving

In Figure 4. Shows the Flow Chart of Data Retrieving. Here describes user A requests to retrieve a message from storage servers. The message is either stored by him or forwarded to him. User A sends a retrieval request to key servers. Upon receiving the retrieval request and executing a

proper authentication process with user A, each key server K Si requests u randomly chosen storage servers to get code word symbols and does partial decryption on the received code word symbols by using the key share SK Ai. Finally, user A combines the partially decrypted code word symbols to obtain the original message M.

5. Data reliability Algorithm

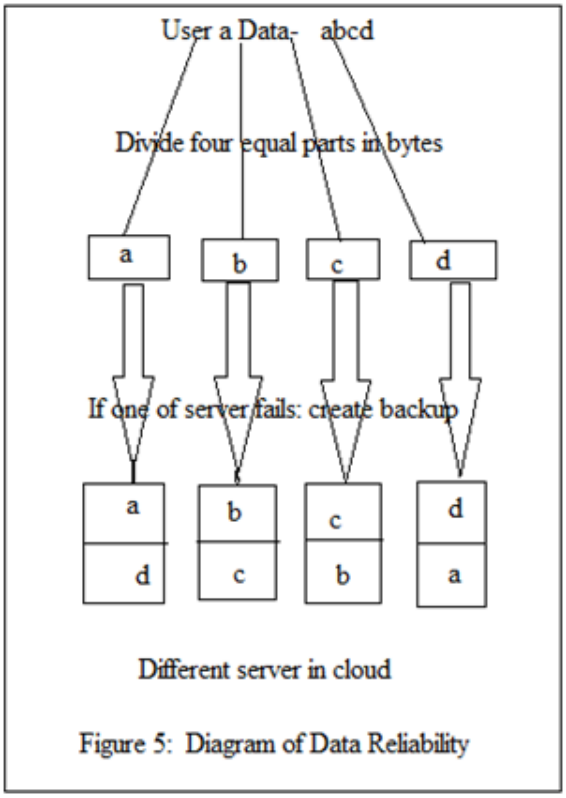


Figure 5: Diagram of Data Reliability

In Figure5 shows the Diagram of Data Reliability. Here describes the algorithm of creating backup of each file part and stored into server due to which if one of the server gets fail due to back up got our complete file.

VI. CONCLUSION

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. Constructing a secure storage system that supports multiple functions is challenging when the storage system is distributed and has no central authority. By using a threshold proxy re-encryption scheme and integrate it with a decentralized erasure code such that a secure distributed storage system is formulated. The distributed storage system not only supports secure and robust data storage and retrieval, but also lets a user forward his data in the storage servers to another user without retrieving the data back. The proxy re-encryption scheme supports encoding operations over encrypted messages as well as forwarding operations over encoded and encrypted messages and generate re-secrete key. Our method fully integrates encrypting, encoding, and forwarding. User analyses and suggest suitable parameters for the number of divided of a message dispatched to different storage servers and the number of storage servers queried by a key server. These parameters allow more flexible adjustment between the number of storage servers and robustness. The main technical contribution is that if secure storage server is failure by any type attack then same secure storage server can create backup so user can get failure data completely and successfully.

REFERENCES

[1] J. Kubiatowicz, D. Bindel, et.al, "Ocean store: *An Architecture for Global-Scale Persistent Storage," Proc. Ninth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 190-201, 2000.*

[2] P. Druschel and A. Rowstron, *"PAST: A Large-Scale, Persistent Peer-to-Peer Storage Utility", Proc. Eighth Workshop Hot Topics in Operating System (HotOS VIII), pp. 75-80, 2001.*

[3] A. Adiya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R. Wattenhofer, *"Farsite: Federated, Available, and Reliable Storage for an Incompletely Trusted Environment," Proc. Fifth Symp. Operating System Design and Implementation (OSDI), pp. 1-14, 2002.*

[4] A. Haeberlen, A. Mislove, and P. Druschel, *"Glacier: Highly Durable, Decentralized Storage Despite Massive Correlated Failures," Proc. Second Symp. Networked Systems Design and Implementation (NSDI), pp. 143-158, 2005.*

[5] H.-Y. Lin and W.-G. Tzeng, *"A Secure Decentralized Erasure Code for Distributed Network Storage," IEEE Trans. Parallel and Distributed Systems, vol. 21, no. 11, pp. 1586-1594, Nov. 2010.*

[6] D.R. Brownbridge, L.F. Marshall, and B. Randell, *"The Newcastle Connection or Unixes of the World Unite!" Software Practice and Experience, vol. 12, no. 12, pp. 1147-1162, 1982.*

[7] 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.

[8] A.G. Dimakis, V. Prabhakaran, and K. Ramchandran, *"Decentralize Erasure Codes for Distributed Networked Storage," IEEE Trans. Information Theory, vol. 52, no. 6 pp. 2809-2816, June 2006.*

[9] M. Mambo and E. Okamoto, *"Proxy Cryptosystems: Delegation of the Power to Decrypt Cipher texts," IEICE Trans. Fundamentals of Electronics, Comm. and Computer Sciences, vol. E80-A, no. 1, pp. 54-63, 1997.*

[10] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, *"Improve Proxy Re-Encryption Schemes with Applications to Secure Distributed Storage," ACM Trans. Information and System Security, vol. 9, no. 1, pp. 1-30, 2006.*

[11] 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.*

[12] G. Ateniese, K. Benson, and S. Hohenberger, *"Key-Private Proxy Re-Encryption," Proc. Topics in Cryptology (CT-RSA), pp. 279-294, 2009*