I J C T A, 9(13) 2016, pp. 6135-6142
© International Science Press

Data Integrity Protection and Security for Data Using Shuffling Algorithm in Cloud Based Storage

K. Brindha[1], P. Anitha[2] and C. Vinothini[3]

ABSTRACT

Organization and people widely uses cloud storage were they can buy or rent storage capacity to store huge data. Cloud storage offers on-demand data outsourcing services. To protect outsourced data against corruption and providing integrity protection and checking has become critical. Security problem arises when data stored is outsourced to third party cloud storage. Data Integrity Protection (DIP) schema is implemented were data are striped and they are encrypted and sent across multiple servers. Here shuffling algorithm is introduced to improve the security of data. This also provides both integrity and security for data which are stored for long term. Regenerating codes provides fault tolerance which checks the integrity of random subsets of outsourced data and protect against corruption and provide availability of data at low cost. This is implemented in a simple thin cloud storage system. Modification of data can be done remotely without downloading the file. Regenerating code make a trade-off between performance and security which is very important for maintaining efficient cloud storage. Remote data integrity checking is performed for regenerating codes to make the cloud storage trustable.

Keywords: data checking, data striping and shuffling secure cloud storage, integrity protection ,trustable cloud storage.

I. INTRODUCTION

Cloud computing provides the capabilities of storing and processing the data in third party data center. It is gaining popularity because of elasticity and its maintenance cost is low.Security is one of the main problem faced in cloud storage. So protecting the data against corruption and enhancing fault tolerance is essential in cloud storage. Data loss, data modification, loss of control and lack of trust should be prevented for efficient cloud storage. We no need to purchase, manage and maintain expensive hardware. Cloud storage provides long-term archival where the data are read and written rarely. The data which is stored should be the same when it is retrieved.

Security issues arises when the data are outsourced to a third party cloud provider. So protecting the data against corruption and enhancing fault tolerance is essential in cloud storage. It is difficult to verify the integrity of data which is outsourced and maintaining security for data that are accidentally corrupted or compromised by inside and outside attacks. The main use of cloud storage is long term archival where the data are stored for long time period and read rarely, so it is necessary to ensure integrity.

In cloud storage there are huge amount of data that has to be stored and retrieved. So two concepts that are introduced are Proof of Retrievability (POR) [16] and Proof of data Possession (PDP) [3]. This is

[*] PG Scholar, Computer science and Engineering, Dr. N. G. P Institute of Engineering, Coimbatore, Tamil Nadu, India, *E-mail: brindha8892@gmail.com*

[**] Assistant Professor, Computer Science and Engineering, Dr. N. G. P. Institute of Technology, Coimbatore, Tamil Nadu, India, *E-mail: anitha@drngpit.ac.in*

[***] Assistant Professor, Computer Science and Engineering, Dr. N. G. P. Institute of Technology, Coimbatore, Tamil Nadu, India, *E-mail: vinothini@drngpit.ac.in*

proposed to spot check the huge amount of data. POR [16] and PDP [3] are proposed only in single server.MR-PDP [10] and HAIL [4] is proposed to check the integrity for multiple server using replication and erasure coding. Erasure coding has low storage overhead than the replication in same fault tolerance level.

Large-scale storage systems [12], [22], [23] experience disk/sector failures in which some ends in permanent loss of data. In commercial cloud-storage data loss events are found [18], [26]. A small failure rate can result in significant data loss in cloud storage archival [29]. To avoid all these drawbacks regenerating codes [11] are introduced.

Regenerating codes minimize repair traffic. Minimizing the repair traffic was done by reading the whole file and reconstructing it. But this was a very difficult process to follow. To make this simple a set of chunks smaller than the original file is taken from another surviving server and reconstructs only the lost data chunks. In simple words when any unreliable nodes are found, a new node should be created.

It allow a new node to communicate functions of the stored data from the surviving nodes. It reduces the repair bandwidth. There is a fundamental tradeoff between storage and repair bandwidth. It can be achieved at any point in this optimal tradeoff. We can enable integrity check with regenerating codes by preserving repair traffic by HAIL [4] concept. It protects data on per file basis and distributes across different servers. To repair lost data in a failed server one has to access the whole file. This does not support the regenerating code [11] concepts. So we need to design integrity protection with regenerating code.

Thus Data Integrity Protection (DIP) for regenerating codes was introduced. We also implement Functional Minimum-Storage Regenerating (FMSR) codes and finally FMSR-DIP codes is constructed. FMSR-DIP codes are to randomly check the data in a multiserver settings. This is constructed for efficient data integrity and security simultaneously. It targets on long-term archives.

Due to security issues in cloud storage regeneration codes are not helpful. So the data which are stored in server are in encrypted form. When there is any loss of data the server backup will automatically heal the lost data. Only thin cloud interface [27] is only assumed where only read/write functionalities are supported. The contributions of FMSR-DIP are it provides integrity protection, fault tolerance and efficient recovery for cloud based storage.

The paper has the following sections. Section 2 describes about the related works. Section 3 describes about the system model for the proposed system. Section 4 describes the experimental results and section 5 describes conclusion and future work of the paper.

II. RELATED WORKS

We consider the problem of integrity security of static data, which is in long term archival storage systems. The major limitation found in POR [16] and PDP [3] is that they are designed only for single server setting. It detects only the corrupted data and cannot recover the original data. Compact Proof of Retrievability [25] is a data storage center prove to the verifier that all the data stored are the client's data only. Here both efficient and provably secure storage system provided. Cumulus [27] is a efficient filesystem backup system which is specifically designed for thin cloud. It provides only least-common-denominator which is get and put of complete files. A proxy-based storage system for fault-tolerant multiple-cloud storage called NC Cloud is introduced, which achieves cost-effective repair for a permanent single-cloud failure.

Functional Minimum-Storage Regenerating (FMSR) codes are introduced which maintain the same fault tolerance and data redundancy as in traditional erasure codes.

This is a Proxy-based design for multiple-cloud storage. Fig 1 is the normal operation, and Fig 2 is repair operation. When the cloud node 1 fails the proxy regenerates the data to new cloud 5.

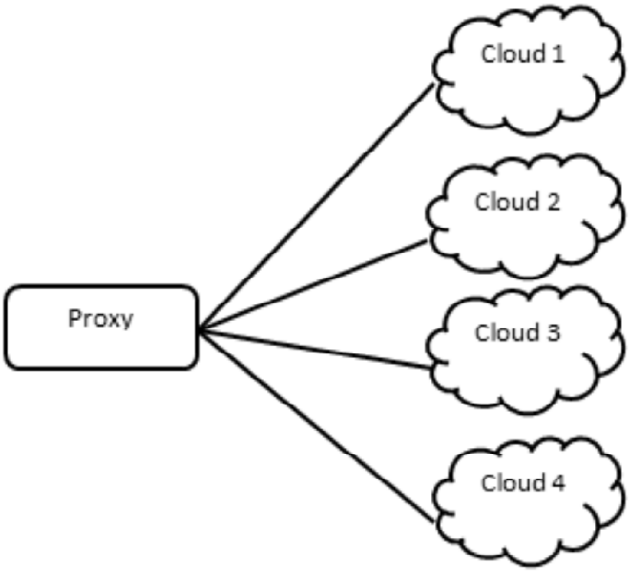

Figure 1: Normal operation

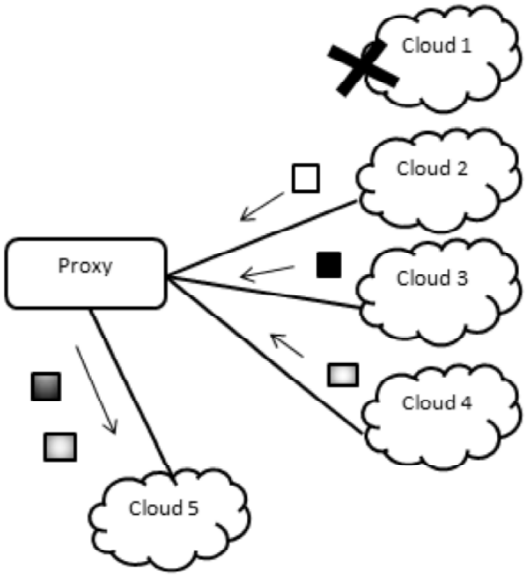

Figure 2: Repair operation

The main objective is to reduce the cost of storage repair for permanent single-cloud failure. Here we compare three codes they are RAID-6, EMSR codes and FMSR codes. This method defines the data being downloaded from the surviving cloud during the single cloud failure recovery. We store a file of size N of four cloud. Each cloud is a logical storage node system.

Now let us consider the RAID-6 codes which is implemented based on the Reed-Solomon codes [31] shown in figure 3(a). The file is divided into two chunks which are considered as native chunks A and B of size N/2. Then two chunks are formed from the native chunks by linear combinations. When node 1 is failed the proxy should download both native chunk and linear combinational chunk. That reconstructs, repair and stores the lost chunk A and that forms a new node A. the total storage size is 2N and the repair traffic is N.

To reduce the repair traffic the regenerating codes are been proposed. Exact Minimum Storage Regenerating (EMSR)[30] keeps the same size of storage as that of RAID-6 codes. But this reduces the repair traffic than the RAID-6 codes as shown in figure 3(b). Here it divides the file into four chunks and when node 1 is down the surviving nodes send the XOR summation of chunks to proxy and reconstructs the lost chunks. The storage size is 2N same that of RAID-6 codes but the repair traffic is 0.7N where 25% is saved.

Functional Minimum-Storage Regenerating (FMSR) are implemented as shown in figure 3(c) where the file is divided into four native chunks and it constructs eight distinct code chunks P1 to P8. Each codes have same size of N/4. Any two chunks can be used to recover the native chunks. If one node is down then the proxy collets one code chunk from surviving nodes and downloads three code chunks of size N/4 the proxy generates P1' and P2'. P1' and P2' are only the linear combination of native chunks, then the proxy writes P1' and P2'to a new node. Here the repair traffic is same that of EMSR codes but the node do not perform encoding during repair.

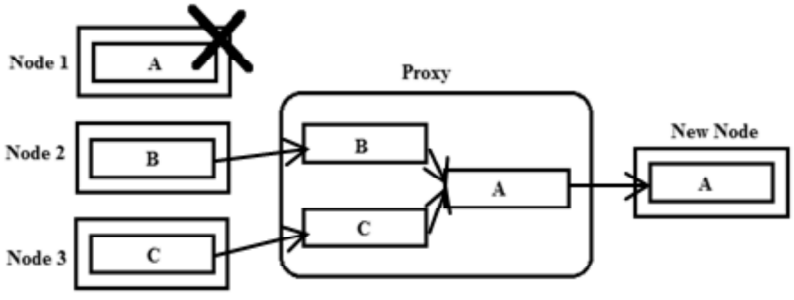

Figure 3(a): RAID-6 codes

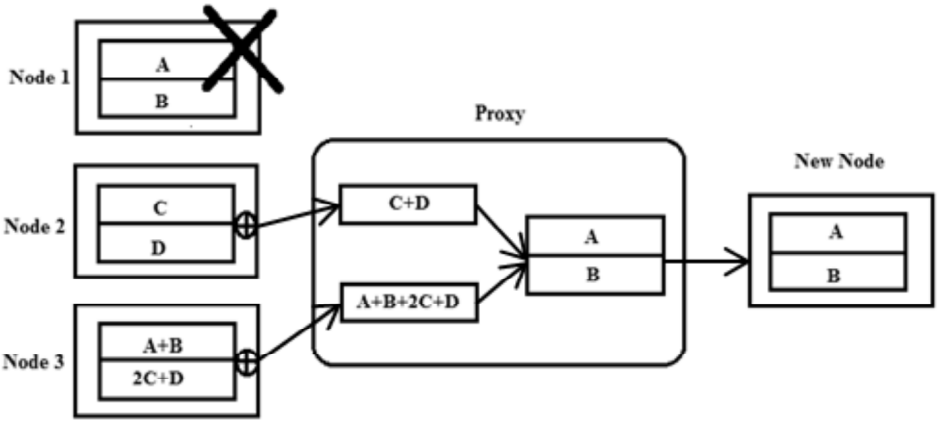

Figure 3(b): EMSR codes

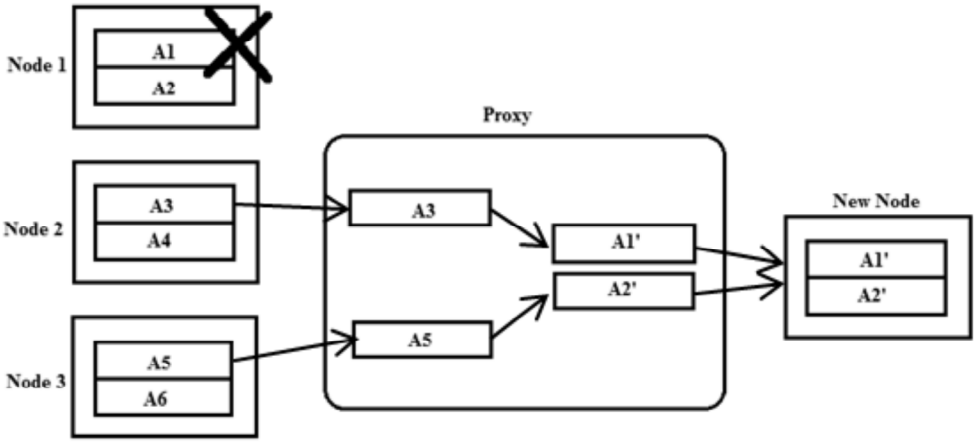

Figure 3(c): FMSR codes

III. SYSTEM MODEL

A cloud based storage system is proposed which is more efficient compared to other storage system. We implement FMSR codes in multiple cloud storage. Here the data which has to be stored is striped and the striped data is shuffled divided in to chunks. Then each shuffled chunks are encrypted using AES algorithm. Those encrypted chunks are sent across multiple cloud storage.

Security is maintained efficiently because the data are shuffled and encrypted and stored in different servers. It is difficult to know in which pattern the data are striped, shuffled and encrypted and sent to different servers. DIP is nothing but it checks that the data which is stored and retrieved are the same. Here remote data integrity checking is done. Data loss, data modification by third party are avoided. The user can store the data and retrieve the data quickly and efficiently. All this process is executed in thin cloud.

The file is uploaded and that file is stripped and shuffled, this represents the FMSR code chunks. That file is then encrypted and that is the FMSR-DIP code chunks. Those chunks is then sent to a storage interface and are distributed to multiple different servers. Fig 4 gives the System Architecture of the above process.

File which are stored and retrieved are performed in private cloud to enhance security. The operations performed are Data Upload, Data checking, Data Retrieval, Data Modification.

(A) Upload operation

The file is uploaded by striping the data and then the those data are shuffled into k(n-k). Then the data are encrypted using AES algorithm into n(n-k) and are sent to storage interface. There the data are distributed

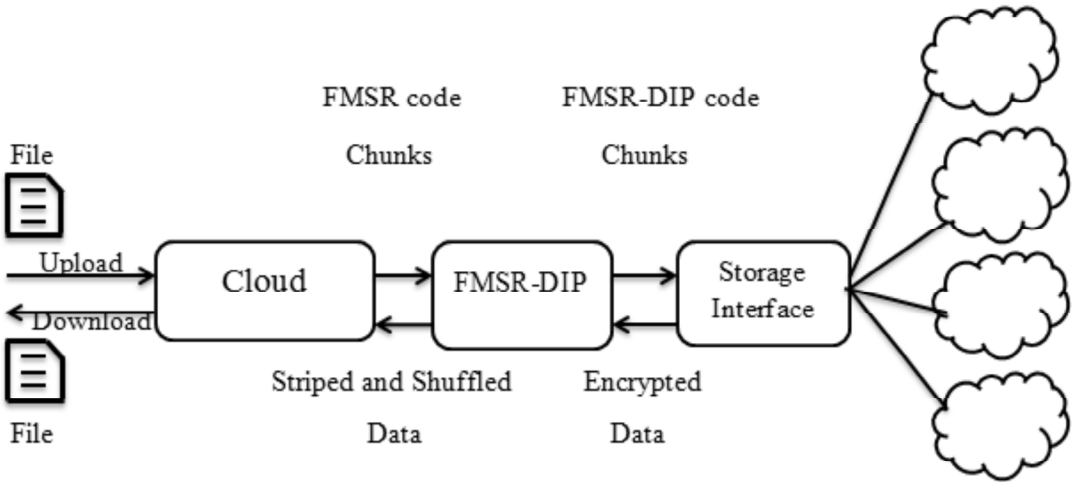

Figure 4 System Architecture

to multiple servers. All the secret key are not revealed to any server, they are kept securely in client side. Then Message Authentication Codes (MAC) are computed for each code chunks. The original FMSR codes only need to download the MAC.

(B) Data checking

Here the checking of data is done to verify the integrity of the data in the server. Remote integrity of data is done here by downloading the copy of encrypted data from each server. As the are encrypted and replicated across server, the majority voting process is done and the corrupted data are resolved. Sampling of data and row verification is also performed to check the data in the cloud based storage.

(C) Data Retrieval

In Data Retrieval process the name of the file is given and the file is retrieved. So there the data are collected from the cloud storage and they are decrypted. Then the decrypted data are reshuffled and the they are retrieved. So both integrity and security of data is maintained for efficient cloud storage.

(D) Data Modification

Data Modification is a process where at the time of retrieval if the file has to be modified the modification is done and then the file is uploaded. Remote modification of text file can be done.

A Adversarial Error-Correcting Code (AECC) [5],[9] is used to protect the chunks against corruption. In conventional Error-Correcting Code (ECC) the large encrypted file is broken into smaller strips and then ECC is applied. But AECC uses Pseudorandom Functions (PRFs) [13], [14] which makes infeasible for target to corrupt any particular striped data. FMSR codes and AECC provides fault tolerance. The main difference between FMSR codes and AECC is FMSR codes is applied for striped data which is stored in server and AECC is applied for single chunk stored in a server. Both are used to improve the integrity and security.

IV. EXPERIMENTAL RESULT

We use thin cloud storage where each server only provides a basic interface for clients to read and write stored files. Now a days cloud storage providers provide a interface called RESTful interface which has only PUT and GET commands. PUT is used to write the file and GET allows to read the file. DIP uses PUT and GET commands to interact with servers.

If the data are corrupted that can be recovered by AECC. The AECC first checks which part of the data is corrupted. The it checks the backup server and heals the encrypted data lost in server. So automatic healing of data loss and corruption is provided to increase the availability of data in cloud storage.

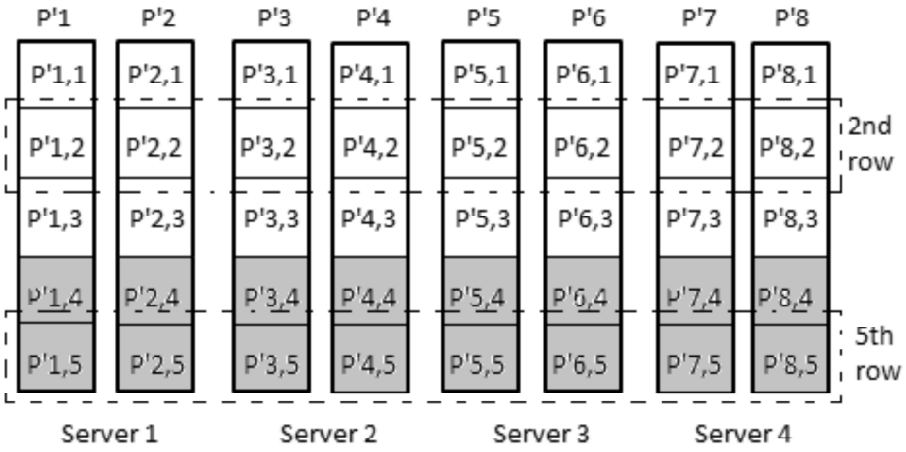

Figure 5: Server Setup

The encrypted data are stored in P'i,1 , P'i,2 and P'i,3. This is how the data are divided and are stored as chunks. P'i,4 and P'i,5 corrosponds to the AECC parities of the chunks. When the data are lost or corrupted the the AECC goes and checks with the backup data and recover the data.

A text file is uploaded by the user. The uploaded text is first sliced. Then the sliced text is shuffled where each character is numbered from 0 to 2 and each numbered characters is formed into grouped of three different chunks. The chunks are encrypted using AES algorithm. The three Chunks are sent to three different servers. During the download process the files is first retrived from the server then it is decrypted, reshuffled, and the file is displayed.

The below graph shows the time taken for uploading and retrieving the data.

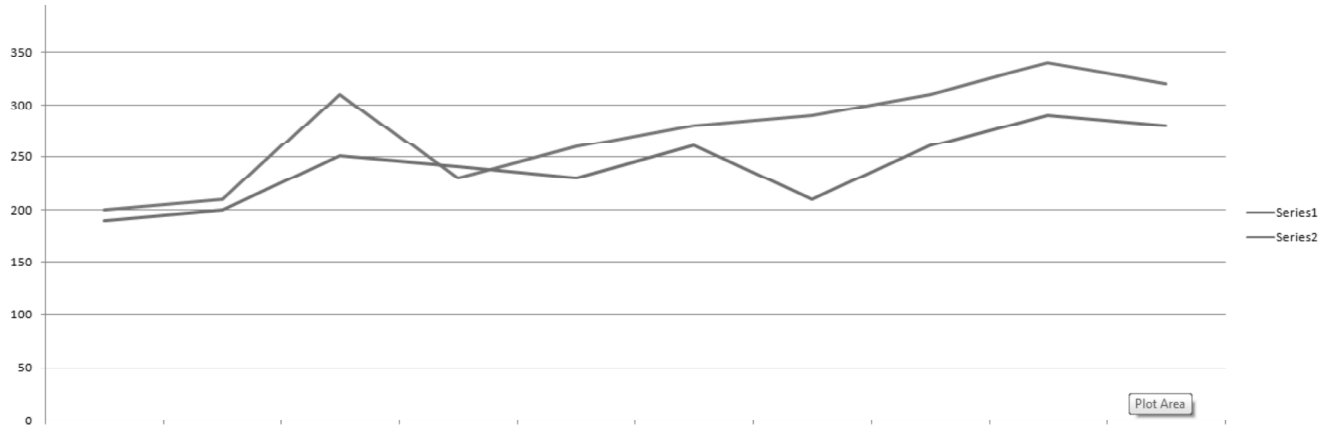

Figure 6: Time taken to upload and retrive the data

When the number of users increases then the time taken to upload and retrive the data efficiency also increases. The time taken to store and retrieve is less compared to the existing system. So the corruption of data is not passible by third party.

The integrity of data is increased than that of the existing cloud storage system because of the regenerating codes. The security of the data are increased where the chunks increase the security by 2% and shuffling method increase security by 6%. So the security of the data in cloud storage is increased by 8%.

V. CONCLUSION AND FUTURE WORK

In this work, we have discussed how security, integrity and availability of data is maintained highly and efficiently in cloud storage. To achieve this we have designed FMSR-DIP codes which shows a trade between performance and security. Data loss and data corruption is managed using AECC, adding fault tolerance to the cloud based storage. We have implemented only for text documents.

In future work, the implement can be done for images. Each cell in the image is stripped and are shuffled and sent to multiple servers. Even for video files this can be implemented, where each individual frame is taken and are stripped, shuffled and are stored in multiple server.

REFERENCES

[1] H. Abu-Libdeh, L. Princehouse, and H. Weatherspoon, "RACS: A Case for Cloud Storage Diversity," Proc. First ACM Symp. Cloud Computing (SoCC '10), 2010.

[2] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," Comm. ACM, vol. 53, no. 4, pp 50-58, 2010.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote Data Checking Using Provable Data Possession," ACM Trans. Information and System Security, vol. 14, article 12, May 2011.

[4] K. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," Proc. 16th ACM Conf. Computer and Comm. Security (CCS '09).

[5] K. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," Proc. ACM Workshop Cloud Comput- ing Security (CCSW '09), 2009.

[6] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-Based Distributed Storage Sys- tems," Proc. ACM Workshop Cloud Computing Security (CCSW '10), 2010

[7] H.C.H. Chen and P.P.C. Lee, "Enabling Data Integrity Protection in Regenerating-Coding-Based Cloud Storage," Proc. IEEE 31st Symp. Reliable Distributed Systems (SRDS '12), 2012.

[8] L. Chen, "NIST Special Publication 800-108," Recommendation for Key Derivation Using Pseudorandom Functions (Revised), http:// csrc.nist.gov/publications/nistpubs/800-108/sp800-108.pdf, Oct. 2009.

[9] R. Curtmola, O. Khan, and R. Burns, "Robust Remote Data Checking," Proc. ACM Fourth Int'l Workshop Storage Security and Survivability (StorageSS '08), 2008.

[10] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-Replica Provable Data Possession," Proc. IEEE 28th Int'l Conf. Distributed Computing Systems (ICDCS '08), 2008.

[11] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network Coding for Distributed Storage Sys- tems," IEEE Trans. Information Theory, vol. 56, no. 9, 4539-4551, Sept. 2010.

[12] D. Ford, F. Labelle, F.I. Popovici, M. Stokel, V.-A. Truong, L. Barroso, C. Grimes, and S. Quinlan, "Availability in Globally Distributed Storage Systems," Proc. Ninth USENIX Symp. Operating Systems Design and Implementation (OSDI '10), Oct. 2010.

[13] O. Goldreich, Foundations of Cryptography: Basic Tools. Cambridge Univ. Press, 2001.

[14] O. Goldreich, Foundations of Cryptography: Basic Applications. Cambridge Univ. Press, 2004.

[15] Y. Hu, H. Chen, P. Lee, and Y. Tang, "NCCloud: Applying Network Coding for the Storage Repair in a Cloud-of-Clouds," Proc. 10th USENIX Conf. File and Storage Technologies (FAST '12), 2012.

[16] A. Juels and B. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," Proc. 14th ACM Conf. Computer and Comm. Security (CCS '07), 2007.

[17] H. Krawczyk, "Cryptographic Extraction and Key Derivation: The HKDF Scheme," Proc. 30th Ann. Conf. Advances in Cryptology (CRYPTO '10), 2010.

[18] E. Naone, "Are We Safeguarding Social Data?" http:// www.technologyreview.com/blog/editors/22924/, Feb. 2009.

[19] J.S. Plank, "A Tutorial on Reed-Solomon Coding for Fault- Tolerance in RAID-Like Systems," Software - Practice & Experience, vol. 27, no. 9, pp. 995-1012, Sept. 1997.

[20] M.O. Rabin, "Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance," J. ACM, vol. 36, no. 2, pp. 335- 348, Apr. 1989.

[21] I. Reed and G. Solomon, "Polynomial Codes over Certain Finite Fields," J. Soc. Industrial and Applied Math., vol. 8, no. 2, pp. 300- 304, 1960.

[22] B. Schroeder, S. Damouras, and P. Gill, "Understanding Latent Sector Errors and How to Protect against Them," Proc. USENIX Conf. File and Storage Technologies (FAST '10), Feb. 2010.

[23] B. Schroeder and G.A. Gibson, "Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You?" Proc. Fifth USENIX Conf. File and Storage Technologies (FAST '07), Feb. 2007.

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

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

[26] "TechCrunch," Online Backup Company Carbonite Loses Customers' Data, Blames and Sues Suppliers, http:// techcrunch.com/2009/03/ 23/online-backup-company-carbonite-loses-customers-data- blames-and-sues-suppliers/, Mar. 2009.

[27] M. Vrable, S. Savage, and G. Voelker, "Cumulus: Filesystem Backup to the Cloud," Proc. USENIX Conf. File and Storage Technologies (FAST), 2009.

[28] "Watson Hall Ltd," UK Data Retention Requirements, https:// www.watsonhall.com/resources/downloads/paper-uk-data-retention-requirements.pdf, 2009.

[29] A. Wildani, T.J.E. Schwarz, E.L. Miller, and D.D. Long, "Protecting Against Rare Event Failures in Archival Systems," Proc. IEEE Int'l Symp. Modeling, Analysis and Simulation Computer and Telecomm. Systems (MASCOTS '09), 2009.

[30] C. Suh and K. Ramchandran. Exact-Repair MDS Code Construction using Interference Alignment. IEEE Trans. on Information Theory, 57(3):1425–1442, Mar 2011.

[31] I. Reed and G. Solomon. Polynomial Codes over Certain Finite Fields. Journal of the Society for Industrial and Applied Mathematics,8(2):300–304,1960.