A Novel Shorten Erasure Based Reed Solomon Fault Tolerance Code in Digital Format for Commercial Cloud Storage

Ramprakash Kota [1], Dr.Rajasekhara Rao Kurra [2]

1 Senior System Architect, USA, Research Scholar, Department of CSE, ANU, India.

2 Director, Sri Prakash College of Engineering (SPCE), Tuni, India.

Abstract— With the continuous growth in the cloud based applications and storage requirements on cloud, many commercial cloud based storage service providers are making the market more competitive with the advanced technologies and reduced cost. The advancement in technologies and high demand for fault tolerant storage solutions most of the cloud based commercial storage service providers are now equipped with Erasure based Reed – Solomon fault tolerance mechanism. However the additional cost for replication is still an overhead for service providers and customers. In this work, we propose a novel erasure based code and further optimization as shortening the proposed code also for the digital storage formats. The work also results into a comparative study of cost analysis for commercial cloud based storage service providers. Finally the work demonstrates the improvement in code shortening and making the performance higher.

Keywords: Erasure, Reed – Solomon, Code Shortening, Performance Comparison, Evolution Application, Response Time Comparison, Dropbox, Google Drive, Hightail, OneDrive, SugarSync

I. Introduction

In the past years, the high upcoming demand for storage with high performance and reliability were been understood. The industry was approaching towards a phase where the lack of standardization of digital storage was limiting the applications to make storage more reliable for commercial storage providers. The major bottleneck for the standardization was the non-standard storage solutions used by different service providers. In the early 80's, the industry adopted cloud computing for distributed storage solutions. The effort was well recognized and multiple companies came together to form a consortium in order to frame the standardization for digital storage.

As far as data storage is concerned, there are multiple schemes are available to improve file and data compression. The other most influencing parameters For instance, a data file that is uploaded and accessed on the server may seriously be effected by the network bandwidth as well as the server workload. This will degrade the efficiency [1]. Moreover the cloud storage services deals with a great scope and domain of the data being storage and retrieved along with the frequency of access varying depending on the mode of the operation performed on the data [2]. Offering unlimited storage container space might cause a high economic drawback on the cloud storage provider and as well as the users due to inefficient storage [3]. Hence, a technique or automation is needed to find the best suitable storage structure based on cost and other influencing factors. There are many free offerings of the cloud storage services; however they may not suite the application requirement to the best always [4].

Two major companies, Philips and Sony took the major initiative to define the standard storage formats in digital media. The standard is well accepted today and been referred as compact storage format. This standard format is majorly used for achieving any data, which also reduces the storage cost compared to the early storage formats. However the compact storage format has limitations in order to achieve high availability. It is difficult to predict how a storage media gets corrupted. In the earlier studies we have understood the reasons for storage device failure. Henceforth we realise the following errors for storage failures as

(1) The additional noise affecting the storage during transmission or during retrieval And

(2) Mishandling of the removable devices

The most important improvement in the recent time for fault tolerance in digital media storages is the Reed – Solomon code. The basic benefit of the Reed – Solomon codes is to rearrange so that the timely restoration can be achieved for storage devices. Thus in this work we concentrate on further enhancement of the Erasure based fault tolerance mechanism.

The rest of the work is framed such as in Section II we understand the cost effectiveness of the commercial cloud storage solutions, in Section III we realise the basic Reed

Solomon Fault Tolerance scheme, in Section IV we propose the novel Reed – Solomon based code, in Section V we propose the further optimization of the proposed code, in Section VI we discuss the implementation and results and in Section VII we conclude the work.

II. Commercial Cloud Storage Services

As the choice of storage services from cloud is not limited and most of those are configured to give best advantages for specific type of data and operation, we compare most of the services here [5 – 7].

A. *Dropbox*

The Dropbox is a storage service which is available for client side access for Windows systems, Linux Systems, Macintosh systems, Blackberry mobile operating systems, Android mobile operation systems and finally the IPhone operating systems. The free Basic account comes with a paltry 2GB of storage. For document based applications this is huge. The Storage service is good choice for applications using the container for read only data.

Table 1. Cost Comparison for Dropbox.

Data Load	Cost
Load in GigaBytes	Price in US Dollars
100	**99 USD**
200	**99 USD**
300	**99 USD**
400	**499 USD**
500	**499 USD**
1000	**Not Available**
> 1000	**Not Available**

Here we provide a graphical representation of the cost price comparison:

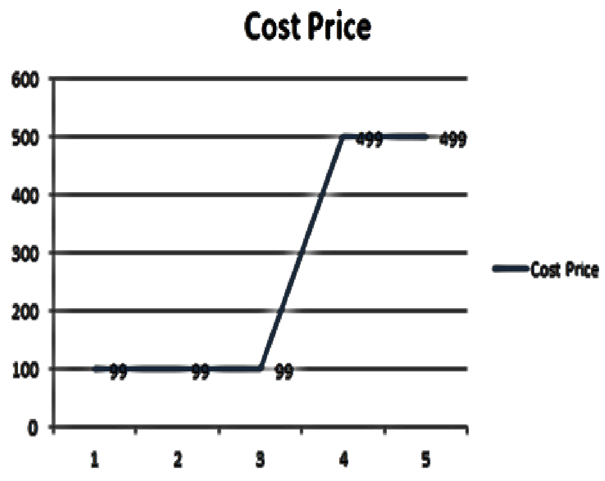

***Fig.1.** Cost Comparison for Dropbox*

Table 2. Support for Mobile Based Cloud Applications in Dropbox

Client OS Type	Support
Apple IPhone Operating Systems	Available
Android Mobile Operating Systems	Available
Blackberry Operating Systems	Available
Microsoft Mobile Operating System	Available

B. *Google Drive*

The most popular cloud storage service is Drive storage from Google. The basic account comes with 15 Giga bytes of storage for a new customer account or an existing account created with Google Email. The highest rated benefit of the Google Drive is the service can be also be integrated with other existing google services for storing various types of data from other services.

Table 3. Cost Comparison for Google Drive

Data Load	Cost
Load in Giga Bytes	Price in US Dollars
100	60 USD
200	120 USD
300	120 USD
400	240 USD
500	240 USD
1000	600 USD
> 1000	1200 to 9600 USD

Here we provide a graphical representation of the cost price comparison:

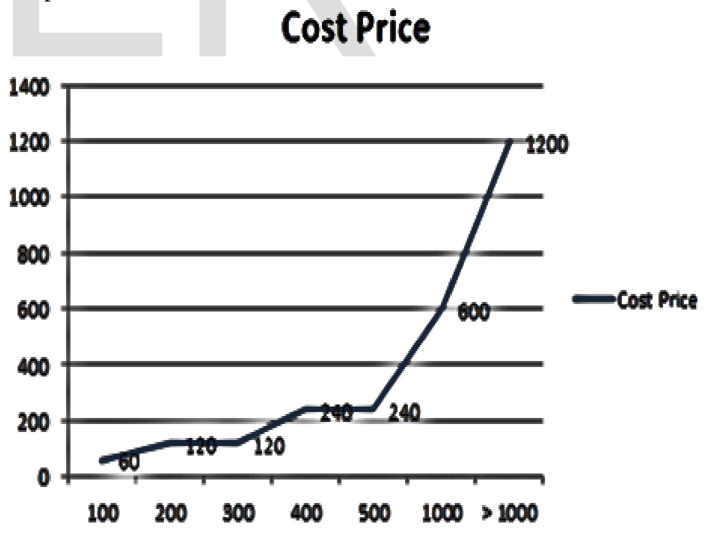

Fig.2. Cost Comparison for Google Drive

Table 4. Support for Mobile Based Cloud Applications in Google Drive

Client OS Type	Support
Apple IPhone Operating	Available

Systems	
Android Mobile Operating Systems	Available
Blackberry Operating Systems	Not Available
Microsoft Mobile Operating System	Not Available

C. Hightail

The previous version of business cloud storage of Hightail was popular by name of YouSendIt. The basic reason for creating the name was the core of the features that Hightail provides. Hightail is majorly known for sharing files, which can be digitally signed for verifications. The core technology behind this provider is link sharing, where the sender can upload a file and the link to that same file can be shared with the recipient. The recipient can click on the link to download the same. This service is popular for business users as it provides the private cloud storage and the desktop version of the client, which can be used for syncing local files to the cloud storage.

Table 5. Cost Comparison for Hightail

Data Load	Cost
Load in Giga Bytes	Price in US Dollars
100	Free
200	Free
300	Free
400	Free
500	Free
1000	Free
> 1000	195 USD

Table 6. Support for Mobile Based Cloud Applications in Hightail

Client OS Type	Support
Apple IPhone Operating Systems	Available
Android Mobile Operating Systems	Not Available
Blackberry Operating Systems	Not Available
Microsoft Mobile Operating System	Not Available

D. OneDrive

The OneDrive was previously popular as SkyDrive. The functionalities are mostly same as Dropbox. The most important factor for this storage service is that the client version is available for Windows systems, Linux Systems, Macintosh systems, Blackberry mobile operating systems, Android mobile operation systems and finally the IPhone operating systems. Moreover the supports for social media plug-ins are also available here. This feature makes the application more compatible with other applications to access data directly.

Table 7. Cost Comparison for OneDrive

Data Load	Cost
Load in Giga Bytes	Price in US Dollars
100	50 USD
200	100 USD
300	Not Available
400	Not Available
500	Not Available
1000	Not Available
> 1000	Not Available

Here we provide a graphical representation of the cost price comparison:

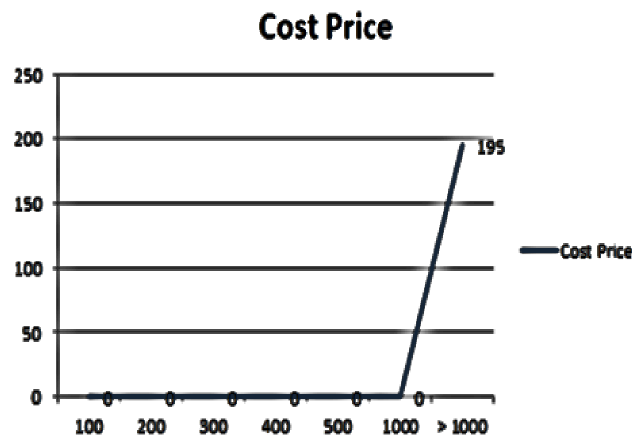

Fig.3. Cost Comparison One Drive

Table 8. Support for Mobile Based Cloud Applications in OneDrive

Client OS Type	Support
Apple IPhone Operating Systems	Available
Android Mobile Operating Systems	Available
Blackberry Operating Systems	Available
Microsoft Mobile Operating System	Available

E. SugarSync

The SugarSync is majorly popular among business users for its effective and fast online backup solutions. The service can also be used for complete folder and individual file syncing with multiple applications and multiple users. Moreover the service provides a unique function to share the stored content over multiple devices at same point of time but with different permission levels. The most important factor for this storage service is that the client version is available for Android mobile operation systems and also the IPhone operating systems.

Table 9. Cost Comparison for SugerSync

Data Load	Cost
Load in Giga Bytes	Price in US Dollars

100	99 USD
200	250 USD
300	250 USD
400	250 USD
500	250 USD
1000	550 USD
> 1000	Pay Per Use

Here we provide a graphical representation of the cost price comparison:

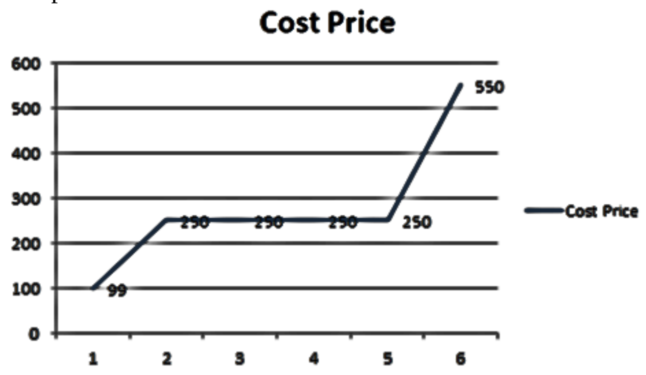

Fig.4. Cost Comparison for Sugar Sync

Table 10. Support for Mobile Based Cloud Applications in SugerSync

Client OS Type	Support
Apple IPhone Operating Systems	Available
Android Mobile Operating Systems	Available
Blackberry Operating Systems	Available
Microsoft Mobile Operating System	Available

III. Reed – Solomon Code For Fault Tolerance

The most important factor that makes Reed-Solomon framework to implement is the simplicity. Here in this work we consider the scenario to compare the performance of Reed – Solomon and Proposed Encoding technique [8].

We consider there will be K storage devices each hold n bytes of data such that,

$$D = \sum D_1, D_2 . D_3 ..... D_k \qquad \text{…Eq 1}$$

Where D is the collection of storage devices

Also there will be L storage devices each hold n bytes of check sum data such that,

$$C = \sum C_1, C_2, C_3 .... C_L \qquad \text{…Eq 2}$$

Where C is the collection of Checksum devices

The checksum devices will hold the calculated values from each respective data storage devices.

The goal is to restore the values if any device from the C collection fails using the non – failed devices.

The Reed – Solomon deploys a function G in order to calculate the checksum content for every device in C. Here for this study we understand the example of the calculation with the values as K = 8 and L = 2 for the devices $C_1$and C_2 with G_1 and G_2 respectively [9].

The core functionalities of Reed – Solomon is to break the collection of storage devices in number of words [10] [11]. Here in this example we understand the each number of words is of u bits randomly. Hence the words in each device can be assumed as v, where v is defined as

$$v = (nbytes).\left(\frac{8bits}{byte}\right).\left(\frac{1word}{u\,Bits}\right) \qquad \text{… Eq 3}$$

Furthermore, v is defined as

$$V = \frac{8n}{u} \qquad \text{…Eq 4}$$

Henceforth, we understand the formulation for checksum for each storage device as

$$C_i = W_i.(D_1, D_2, D_3 ... D_k) \qquad \text{…Eq 5}$$

Where the coding function W is defined to operate on each word

After the detail understanding of the Erasure fault tolerance scheme, we have identified the limitations of the applicability to the cloud storage services and propose the novel scheme for fault tolerance in this work in the next section.

IV. Proposed Novel Fault Tolerance Scheme

With the understanding of the limitations of existing erasure codes to be applied on the cloud based storage systems as the complex calculations with erasure codes will reduce the performance of availability measures significantly. Thus we make an attempt to reduce the calculation complexities with simple mathematical operations in the standard erasure scheme.

The checksum for storage devices are considered as C_i from the Eq 5. We propose the enhancement as the following formulation for checksum calculation:

$$C_i = W_i.(D_1, D_2, D_3 ... D_k) = W_i(D_1 \oplus D_2 \oplus D_3 ... \oplus D_k) \qquad \text{…Eq 6}$$

Here the XOR operation being the standard mathematical operation most suitable for logical circuits used in all standard hardware makes it faster to be calculated.

Also we redefine the function to be applied on each word for the storage devices D as following:

$$W = \begin{bmatrix} w_{1,1} & . & . & . & w_{1,L} \\ . & . & . & . & . \\ . & . & . & . & . \\ . & . & . & . & . \\ w_{K,1} & . & . & . & w_{K,L} \end{bmatrix}_{K\,X\,L} \quad \text{…Eq 7}$$

The proposed matrix will be stored on one of the devices and will be recalculated only once. As the modified checksum formulation is an XOR operation, thus which will automatically notify in case of any change.

Furthermore, we optimize the proposed code framework in the next section.

V. Optimizing Proposed Novel Fault Tolerance Scheme

The Reed Solomon code is expressed by the power of coefficient denoted by n for the data blocks, where n is expressed as

$$n = 2^m - 1 \quad \text{…Eq 8}$$

and the code blocks are represented as

$$k = 2^m - 1 - 2t \quad \text{…Eq 9}$$

Where m represents the number of bits per data and t represents the capability of correcting errors. In general the Reed – Solomon code considers an 8 bit data and 2 bit code, the error correcting code can be represented as (255,251) code.

Here in this part of the work, we try to optimize the code length further to reduce the replication cost. The steps of the optimization algorithm are explained here:

Step-1. First we consider the effective code in (255,251) block, where the code is consisting of zero and non-zero codes.

Step-2. Then we find the number of zero codes in the segment. For instance the numbers of zero codes are 227 in the code block. These codes will not have any effect in the error correction and fault tolerance mechanism.

Step-3. Then we find the effective block of the code as (28,24) for a 2 bit error correction code.

Step-4. Hence as a final outcome of the optimization technique, we got the optimized code block.

VI. Implementation and Results

To simulate and understand the improvement in the outcomes we implement the Reed – Solomon code with the enhancement and optimization proposed in this work.

We accept any random data as the initial data block for the testing [Table -11].

Table 11. Initial Data Block

0	0	0	0
1	0	0	0
0	1	0	0
0	0	1	0
0	0	0	1
1	1	0	0
0	1	1	0
0	0	1	1
1	1	0	1
1	0	1	0
0	1	0	1
1	1	1	0
0	1	1	1
1	1	1	1
1	0	1	1
1	0	0	1

Based on the modified fault tolerance scheme, we realise the addition and multiplication table [Table -12 & 13].

Table 12. Addition Table

	0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14
0	0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14
a^0	a^0	0	a^4	a^8	a^14	a^1	a^10	a^13	a^9	a^2	a^7	a^5	a^12	a^11	a^6	a^3
a^1	a^1	a^4	0	a^5	a^9	a^0	a^2	a^11	a^14	a^10	a^3	a^8	a^6	a^13	a^12	a^7
a^2	a^2	a^8	a^5	0	a^6	a^10	a^1	a^3	a^12	a^0	a^11	a^4	a^9	a^7	a^14	a^13
a^3	a^3	a^14	a^9	a^6	0	a^7	a^11	a^2	a^4	a^13	a^1	a^12	a^5	a^10	a^8	a^0
a^4	a^4	a^1	a^0	a^10	a^7	0	a^8	a^12	a^3	a^5	a^14	a^2	a^13	a^6	a^11	a^9
a^5	a^5	a^10	a^2	a^1	a^11	a^8	0	a^9	a^13	a^4	a^6	a^0	a^3	a^14	a^7	a^12
a^6	a^6	a^13	a^11	a^3	a^2	a^12	a^9	0	a^10	a^14	a^5	a^7	a^1	a^4	a^0	a^8
a^7	a^7	a^9	a^14	a^12	a^4	a^3	a^13	a^10	0	a^11	a^0	a^6	a^8	a^2	a^5	a^1
a^8	a^8	a^2	a^10	a^0	a^13	a^5	a^4	a^14	a^11	0	a^12	a^1	a^7	a^9	a^3	a^6
a^9	a^9	a^7	a^3	a^11	a^1	a^14	a^6	a^5	a^0	a^12	0	a^13	a^2	a^8	a^10	a^4
a^10	a^10	a^5	a^8	a^4	a^12	a^2	a^0	a^7	a^6	a^1	a^13	0	a^14	a^3	a^9	a^11
a^11	a^11	a^12	a^6	a^9	a^5	a^13	a^3	a^1	a^8	a^7	a^2	a^14	0	a^0	a^4	a^10
a^12	a^12	a^11	a^13	a^7	a^10	a^6	a^14	a^4	a^2	a^9	a^8	a^3	a^0	0	a^1	a^5
a^13	a^13	a^6	a^12	a^14	a^8	a^11	a^7	a^0	a^5	a^3	a^10	a^9	a^4	a^1	0	a^2
a^14	a^14	a^3	a^7	a^13	a^0	a^9	a^12	a^8	a^1	a^6	a^4	a^11	a^10	a^5	a^2	0

Table 13. Multiplication Table

	0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
a^0	0	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14
a^1	0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0
a^2	0	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1
a^3	0	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2
a^4	0	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3
a^5	0	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4
a^6	0	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5
a^7	0	a^7	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6
a^8	0	a^8	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7
a^9	0	a^9	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8
a^10	0	a^10	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9
a^11	0	a^11	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10
a^12	0	a^12	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11
a^13	0	a^13	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12
a^14	0	a^14	a^0	a^1	a^2	a^3	a^4	a^5	a^6	a^7	a^8	a^9	a^10	a^11	a^12	a^13

Henceforth, we compare the results of the generic Reed-Solomon Coding and the proposed fault tolerance technique [Table – 14] based on the initial code.

Table 14. MULTIPLICATION TABLE

Parameter	Generic RS	Proposed Optimized RS
Initial Polynomial	a^1 a^3 a^5	a^1 a^3 a^5
Encoded Data	a^5 a^3 a^1 a^6 a^4 a^2 a^0	0 0 0 a^6 a^4 a^2 a^0
Fault Tolerance Code	a^5 a^3 a^1 a^6 a^4 a^2 1	a^6 a^4 a^2 1
Optimization Reduction	0%	57%

VII. CONCLUSION

In this work the commercial cloud storage services are been compared based on the cost and performance factors. The result of the comparative measures provided the understanding of the demand for highly reliable and cost effective fault tolerance system. Henceforth, in this work we study the core Reed - Solomon fault tolerance mechanism based on Erasure codes. The work contributes towards the improved performance code for fault tolerance for digital storage devices rather than magnetic. Also the work enhanced the performance of the proposed technique by applying the improvement in terms of optimization. The result of the proposed optimization technique is 57% reduction in the storage cost without negotiating with the fault tolerance reliability.

REFERENCES

[1] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, “Oceanstore: 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. Adya, 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] 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.

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

[7] 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.

[8] J. Bellorado and A. Kavcic , "Low-complexity soft-decoding algorithms for Reed-Solomon codes: An algebraic soft-in hard-out chase decoder" , IEEE Trans. Inf. Theory. , vol. 56 , no. 3 , pp.945 -959 , 2010

[9] F. Garc-Herrero, J. Valls and P. K. Meher , "High-speed RS (255, 239) decoder based on LCC decoding" , Circuits Syst. Signal Process. , vol. 30 , no. 6 , pp.1643 -1669 , 2011

[10] W. Zhang, H. Wang and B. Pan , "Reduced-complexity LCC Reed-Solomon decoder based on unified syndrome computation" , IEEE Trans. Very Large Scale Integr. (VLSI) Syst. , vol. 21 , no. 5 , pp.974 -978 , 2013

[11] J. Jiang and K. R. Narayanan , "Algebraic soft-decision decoding of Reed-Solomon codes using bit-level soft information" , IEEE Trans. Inf. Theory , vol. 54 , no. 9 , pp.2008 -3928

ABOUT THE AUTHORS

Mr.RamPrakash Kota is a Senior System Architect working for Medical Client, USA. He worked at BVRIT Narsapur for 3 years as an Assistant Professor in the department of MCA. He worked as an IT Analyst for 4 years for TCS. Currently he Pursuing Ph.D in the area of Cloud Computing from Acharya Nagarjuna University, Guntur, India.

Dr. Rajasekhara Rao Kurra, Director, Sri Prakash College of Engineering(SPCE), Prof. Kurra Rajasekhara Rao is a Professor of Computer Science & Engineering and presently working as Director, Sri Prakash College of Engineering(SPCE),Tuni. He worked at KLCE/K.L.University for 20 years as a faculty member in various positions as HOD of CSE, HOD of IT,Vice-Principal, Principal, K L College of Engineering (Autonomous), and Dean (Administration), Dean (Faculty & Student Affairs) Dean (Examinations & Evaluation) of KLU. Having more than 28+ years of teaching and research experience, Prof. KRR is actively engaged in the research related to Embedded Systems, Software Engineering and Knowledge Management. He had obtained Ph.D in Computer Science & Engineering from Acharya Nagarjuna University (ANU), Guntur, Andhra Pradesh, India. He is a Life Member of ISCS, IE, ISTE, CSI and IETE. He published more than 80 papers in various International/National Journals, Conferences and produced 4 Ph.D's till now.