Approach to Secure Stored Data in Cloud Computing

A Dissertation submitted in partial fulfillment of the requirement for the

Award of degree of

MASTER OF TECHNOLOGY

IN

INFORMATION SYSTEMS

By

LOMASH GOYAL

Roll No. – 2K11/ISY/12

Under the esteemed guidance of

N.S. RAGHAVA

ASSOCIATE PROFESSOR

Department of Information Technology

Delhi Technological University

2012-2013

CERTIFICATE

This is to certify that the thesis entitled **"Approach to Secure Stored Data in Cloud Computing"** submitted by **Lomash Goyal (2K11/ISY/12)** to the Delhi Technological University, New Delhi for the award of the degree of **Master of Technology** is a bonafide record of research work carried out by him under my supervision.

The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

N.S. Raghava
Associate Professor
Department of Information Technology
Delhi Technological University, Delhi

ACKNOWLEDGEMENT

I feel privileged to offer sincere thanks and deep sense of gratitude to my project guide **N.S. Raghava,** ***Associate Professor,*** for their guidance and constant supervision. I am highly indebted to him for providing me useful and necessary information regarding the project and showing me the right directions in completing the project.

I would like to express my special gratitude and thanks to **Dr. O.P Verma,** ***Head of IT Dept.*** for giving me such an opportunity to work on the project.

This thesis could not have been written without the help and inspiration of my **parents and colleagues** specially Gaurav Chawla and Piyush Khandelwal with whom I have had the pleasure to work.

I am also grateful towards many individuals and the staff of Delhi Technological University for their kind co-operation and encouragement which helped me in completion of this project.

Lomash Goyal
Roll No.: 2K11/ISY/12
Dept. of Information Technology
Delhi Technological University

ABSTRACT

In recent years Cloud Computing has gained ample popularity amongst most of the research oriented institutions and organizations. Cloud computing promises scalability of resources and has on-demand availability of resources. However use of Cloud Computing has posed many challenges regarding the security concerns of the user data. In contrast to the traditional solutions where IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. Cloud computing provides its users a combination of economic and performance benefits, but the security issues that are associated with it can't be neglected. The data stored in the cloud may be frequently updated by the users, using insertion, deletion, modification, appending, reordering, etc. This thesis is focused on cloud data storage security, which has always been an important aspect of quality of service.

To ensure correctness and integrity of users' data on the distributed servers present on cloud, an effective and flexible distributed scheme has been proposed in this dissertation. The proposed scheme makes use of HMAC (Hashed message authentication code) in the form of a homomorphic token and drastically improves data storage correctness and integrity. Further, the implementation of erasure coded techniques, an efficient methodology to recover single or multiple distributed server failure has also been proposed. An analytical comparison has been made between some of the most prominent erasure coding schemes known for accomplishing the above task. On basis of these comparative analyses, Fountain codes are found to give the best results, which justify their use.

This thesis basically aims to provide a standard approach for the storage of data in distributed

architecture which is resilient against any malicious server failure. Use of erasure coded techniques in the data distribution and preparation provides the redundancies and guarantees the data dependability. With the help of homomorphic token(HMAC) along with distributed verification of erasure-coded data, the proposed scheme almost guarantees the storage correctness and simultaneous localization of data errors in case of any data modification or corruption, i.e., the identification of the misbehaving server(s).

LIST OF FIGURES

List of Tables

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

- MOTIVATION
- OBJECTIVE
- CLOUD COMPUTING DEFINED
- LAYERS OF CLOUD COMPUTING
- CLOUD COMPUTING VS OTHER TECHNOLOGIES
- BENEFITS OF CLOUD COMPUTING
- ISSUES WITH CLOUD COMPUTING
- IDENTIFYING CLOUD COMPUTING RISKS
- ORGANIZATION OF THESIS

Cloud computing provides users a combination of economic and performance benefits, but the security issues that are associated with it also can't be neglected. There are a number of examples in the history that justifies the Cloud computing security concerns. Recent downtime of Amazon S3 [1] is one such an example. Moving data into the cloud offers great convenience to users since they don't have to care about the complexities of direct hardware management, but users have to rely on Cloud provider's security measures for the confidentiality and integrity of their data. Confidentiality ensures that data is not disclosed to any unauthorized user and integrity prevents the improper modification of information. For maintaining the proper security it should need to be very much clear with the fact that traditional cryptographic primitives for the purpose of data security protection cannot be directly adopted due to the users' loss control of data under Cloud Computing. Considering various kinds of data for each user stored in the cloud and the demand of long term continuous assurance of their data safety, the problem of verifying correctness of data storage in the cloud becomes even more challenging. In practice, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user's data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature. Enforcing security policy and meeting compliance requirements are tough enough when you deal with third parties and their known or unknown subcontractors, especially on a global scale. User entrust the cloud provider for a safe data warehouse and expects to retrieve its data whenever required. So, there must be certain provision such that user get ensured at regular intervals that his data is securely stored without any alterations and modifications.

Further in case of any distributed server(s) breakdown, there must be certain provisions for retrieving the data stored on it. Although data redundancy is a very obvious solution to this problem, but creating redundant copies of the complete data will lead to wastage of storage resources.

1.1 Objective

Though IT services from the cloud are becoming increasingly in demand around the world, almost every survey and study shows that there are also many concerns which discourage users away from using Cloud Computing services. As more and more information on individuals and companies are placed in the cloud; concerns are beginning to grow about just how safe an environment it is. A lack of faith in the security of the services provided is frequently cited as being one of the main barriers.

The primary objective of this dissertation is to provide a basis of security for the user's data stored on cloud. As a further step, this report also aims to provide a standard approach for the storage of data in distributed architecture which is resilient against any malicious failure. There are already many erasure coded techniques which are being used in case of any server(s) failure(see chapter 2) in the distributed data storage scheme, but this thesis aim to propose the best among those technique on the basis of running time complexity and the implementation cost too. Another key design objective of this study is focused on developing an approach through which user could assure himself that his data is kept safely on the server. The user should challenge the cloud provider whenever desired regarding the verification of stored data.

1.2 Cloud computing defined

Cloud computing is fairly new and has thus no long history. In general it originates from the late nineties and has been further developed in the next millennium, the name was created because the data send couldn't be tracked anymore when moving towards it destination. The term cloud was created because you could not determine the path a certain data package followed. The term cloud computing changed over time. In the early years of cloud computing, the organization Amazon was active in the area of cloud computing. They were already a large organization investing in cloud computing. They had huge data centers which normally only use about 8 to 12% of their computing power. The rest was reserved for whenever peak usage was necessary. They started to use cloud computing in order to save costs in these huge datacenters. After this they were the first to provide cloud computing to the outside world (the customers). This happened in the year of 2006 according to Computer Weekly (2009). Not much later IBM and Google showed interest into cloud computing and started to invest. It seemed that cloud computing showed potential.

In giving a definition to cloud computing, the highest hits on Google scholar are used, in particular the ones with the most references regarding cloud computing. Some scientific papers show up a lot such as [1][2]. In total Google scholar shows about 100 to 120 relevant hits regarding cloud computing. Off course there are a lot more hits, but they go out of the scope of this research. The papers used for this thesis are usually independent which adds some trustworthiness compared with company papers.

As a recapitulation, cloud computing is stated into different definitions. There are some definitions that define a cloud as an updated version of utility computing [3]. The other, and

broader, side states that anything you can access outside your firewall is cloud computing, even outsourcing [4]. This thesis takes the definition in the middle of these two.

In general cloud computing provides hardware and software services that are in the cloud and can be accessed by client as they pay for it. In the "cloud" means that there is no dedicated hardware reserved in a cloud providers' servers.

To get more into detail about cloud computing, the components will be discussed that are used in the clouds. In general there are three main components in cloud computing, these are the servers, the datacentres and the clients [5]. They all connect through the internet with each other and can be seen as a network.

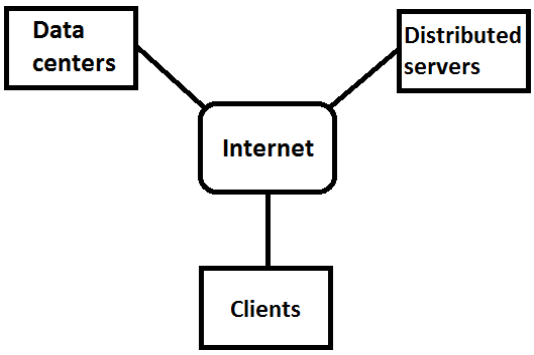

Figure 1.1: Simple Cloud computing network

1.2.1 Datacentres and Distributed servers

In general the data centres contain the services that clients want to obtain whenever they need it. This centre is often a large space which contains all servers providing these services and keeping them up and running. It is also possible to have virtual servers which reduce the amount of actual servers and space [6]. Distributed servers are a name for those servers that are not all in one location. It doesn't matter where these servers are, as a user you won't notice anything different.

These kinds of servers provide high flexibility because it doesn't matter where they stand as long as they are connected to the internet. It is easy for making a back –up of other servers. Besides this, there is no limitation in expanding the cloud [5].

1.2.2 Clients

Most general clients are regular desktop PC's or laptops. Other clients nowadays are also mobile phones (PDA), [7]. The mobile devices are of big importance for cloud computing. They provide the high mobility to those who are trying to access the cloud. In general there are three sorts of clients to distinguish. These are mobile, thin and thick clients. Mobile clients are those with mobile phones [5]. Thin clients are using remote hardware and software. What a user sees is visualized by the server and not by an own hard disk with operating system. On the contrary, thick clients use own hard disks and usually access the cloud trough a web browser.

1.2.3 Users

Logically, behind the clients come the users. Without users, there is no purpose for a cloud. In cloud computing, users can be distinguished four different types of users [5]. All these groups of users will be explained.

The groups to be distinguished as users in cloud computing are:

- Internet Infrastructure developers

- Service Authors

- Integration and provisioning experts

- End users

To point out the differences between the users and for the sake of understanding better what cloud computing is and how it is maintained, all users are explained. Even though for this thesis the focus lies on the end users it is important to distinguish these four kinds of users of the cloud.

1.2.4 Developers

Developers in the cloud are also called as (Internet-infrastructure), Developers are those who develop and maintain the cloud. They have to guarantee and develop that all services get integrated [8]. Their task is to provide end users with a simple interface, and keeping the complexity at a lower level.

1.2.5 Service authors

These authors are somewhat different from the developers but in some cases have overlapping function. Where developers focus on providing all services, authors focus on individual services which may get used directly. Unlike the developers they don't need knowledge about technical specification of the cloud; they solely focus on providing easy to use services [8].

1.2.6 Integration and provisioning experts

These experts are really more focused on the end-user solutions. They are trying to interface with end users, and try to meet in what end users want [5].

1.2.7 End users

The end users eventually have the highest importance as is mentioned before. End users expect that their cloud services have clear and easy to use interfaces, support and information provision. Also the end users have to be protected from any hazard. Therefore it is important to guarantee

security in a cloud, something what will come up later in this thesis. All these requirements make no difference for the kind of users. Some users may hire cloud services for hours, and some for

years. These different end-users should meet the same service as they could have equally important data streams into the cloud. The service also depends upon the Service Level Agreement.

A Service Level Agreement (SLA) is included in a service contract between two parties. This agreement states what services are guaranteed by one party to the other. It states for example the performance agreements, but also more importantly the security and safety agreements.

1.3 Layers

Cloud computing consist of several layers. These three layers represent the SaaS (software), PaaS (Platform) and IaaS (infrastructure). All three abbreviations end with … as a Service, meaning that all these layers provide some kind of services to end users [9],

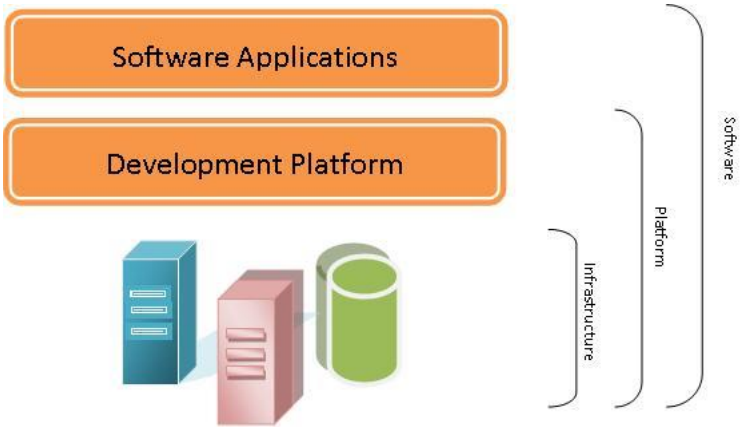

Figure 1.2: Overview of layers in cloud computing: Demystifying SaaS, PaaS and IaaS. (E2E networks 2012)

The image summarizes the tree layers into one picture. To get more into detail the different layers are explained.

1.3.1 Software as a Service

The name speaks for itself; SaaS provides software as a service. Users can 'rent' the software from the cloud provider and do not have to purchase software including the licenses themselves. The provider sets the software in the cloud to be accessible to those whom access the cloud trough internet [10]. This form of cloud computing is gaining popularity and is also the most used by end users. Think of services from Google; a lot of people use G-mail, so accessing Google docs is not that far away anymore (Google docs is SaaS). This thesis is even partly written in Google docs as this software saves your document in the cloud all the time and can be accessed from anywhere in the world. You simply logon to your account (first register) and you get access to your own little cloud. The server hosts the software, in this case the word processor, and besides this it holds the text documents you write or upload. Google provides all these services (for personal users free), so you do not need to purchase or install a license for a word processing application.

1.3.2 Platform as a Service

PaaS differs from SaaS as this service does not provide all the software needed. This is a sort of platform, or a sort of operating system available on the web. The cloud provider implements scripts from a user and makes it available in the cloud. This platform service gives the possibility to create, test and maintain aps in the cloud. Just like SaaS though, the provider of the cloud services is still responsible for any hardware behind the cloud. Another similarity is the pay-as-you-go system. You pay for the use you make of the cloud. Several types of PaaS can be

distinguished [9]. The best and most familiar example is facebook.com; this is a social application platform. Another example is the huge organization amazon.com which provides cloud computing, this PaaS is called raw compute platform. Other two forms of PaaS are application and business application platforms.

1.3.3 Infrastructure as a Service

The name of IaaS also speaks for itself. In a few simple words it provides an organization with a complete infrastructure for IT. Clients are able to purchase this infrastructure whenever they feel like buying it. It is similar to SaaS and PaaS. Moving the infrastructure to the cloud actually means that an organization moves the hardware to the cloud. The rest of mainframes, servers, databases etc. can be obtained from the cloud.

The image on the previous page probably will be clearer now. The infrastructure is the basics and users actually use the hardware. With platforms users tend to create and develop applications and with software the users tend to use the applications only.

1.4 Cloud computing compared with other technologies

With the understanding of what cloud computing is, we might see some similarities with other technologies. This paragraph is all about explaining what Cloud computing isn't and what the differences are with similar looking technologies. Most of these technologies are older than cloud computing and more familiar with the audience, therefore it is important to distinguish it from Cloud computing. The systems of Autonomic computing are the first to be mixed up with cloud computing. This form of computing differs in the way it works. The goal of autonomic

computing is to provide systems than work autonomous. This means that they have to be able to do self-managing. They must configure and fix failures themselves. It is similar to cloud computing because it also consists of large computer systems that have a high-level guidance from humans.

The difference between cloud computing and grid computing is more refined, but it is easy to explain. Grid computing focuses on large scale whereas cloud computing provides services for both smaller and larger scale. Grid computing usually provides high performance constantly, and (the major advantage of) cloud computing provides the performance when necessary [3].

Another comparison is drawn with mainframes; the difference might be clear with a mainframe, but there also similarities. A mainframe could be seen as a cloud. Though it is clear that a mainframe provides access to employees in large organization and the mainframe is completely centralized. That is what differs with cloud computing, as also is the performance. Mainframes provide continuously high performance and cloud computing only whenever necessary [2].

The comparison also has been drawn with peer-to-peer systems. This is because there is a whole cloud of users which are both “client” and “servers”. This is also the difference. In cloud computing clients themselves do not act as providers of any service.

The last comparison that is discussed is the comparison with service oriented computing. Off course cloud computing is service oriented. But service oriented computing focuses more on techniques that run in the SaaS. Cloud Computing, focuses on providing computing services rather than the techniques.

1.5 Benefits of cloud computing

It is easy to say that cloud computing provides benefits to those who use it. The idea is to find out what these benefits actually are. As mentioned before, the major benefit for any end user is that cloud computing can be used simply whenever you need it. It is a pay-as-you-go system. The question then is: why is this actually a benefit? To begin with the user organization, there is no physical room necessary for all the hardware to install. Furthermore there are no maintenance costs for all the hardware [5].

Besides the hardware it is the applications that provide benefits. The cloud is filled with applications that are ready to use, and more important the data used in this application is always accessible from anywhere in the world.

An SLA (Service Level Agreement) guarantees that quality measures are known before entering a cloud. These SLA's are important for the users and can be better maintained then when an organization purchases all the hard and software by itself.

Not a direct benefit, but also important, is that the datacenters are usually placed at strategic chosen places that lower the costs of maintenance. Think of low wages countries.

Focusing more on the users of the cloud the benefits become more concrete. As has become very clear now, scalability is one of the major benefits. When an organization is expecting a peak in its IT use, they simply acquire more IT services from the cloud. This is also the beauty of it, it is very simple. Because huge organizations have invested in Cloud computing, the users can also expect a certain degree of security [5].

Cloud computing provides thus a combination of economic and performance benefits. The economic benefit lies in the costs that have to be made whenever an organization needs

additional IT services, and this relates to the performance benefits. The extra performance can be acquired whenever necessary and improves the performance of an organization directly

1.6 Issues with cloud computing

New technologies come with risks and unknown factors. Something which isn't different with cloud computing. IT intensive organizations will in essence outsource their processes. Some of them could be part of their core business. With bad security these organizations will be exposed to huge risk as their critical data could get exposed to the outside world. Other issues have to do to with legal and privacy issues.

Legal issues differ per country but in general there is expected that not all organization will be allowed to enter public clouds. This means that there will be an increase in the use of private clouds. In next section discussion of the privacy and technological (or security) issues briefly as they apply more to the costumers.

1.6.1 Security

The cloud computing security issues can in general be divided into seven different categories according to [11]. These risks are from the customer's point of view. Risks for providers are not in the scope of this research and therefore not discussed.

Data is processed outside of the organization. This logically brings a certain amount of risk, because in a sense it is a form of outsourcing. This causes to shift any form of security from the organization to the outsourced organization. It is thus for customers important to be familiar with risk procedures on beforehand. The customer himself is still responsible in the end. The providers

have to meet certain standards in security, but these could be of insufficient level for people who want to do harm to an organization. Therefore it is important to know what procedures the provider follows, and it is also important as a costumer to process the communication between the organization and the provider in a secure way.

As has been explained before, the physical location of the cloud could be anywhere [5]. As a customer you cannot always know where your information is at a certain time. This means that they could have their services running in other countries which have other legal issues. This could result in other security standard for a particular country and jeopardize the organization in the cloud.

There are different organizations in the cloud; they work all along in that same cloud. It is not hard to imagine that when fifty different organizations access the cloud it is probable that data gets interleaved and any specific user is unable to fetch his/her data .

This brings several security issues. For example not knowing where your data physically is stored, what would happen in the case of a natural disaster? The provider should provide a back-up for when such disasters happen. This is something that needs to be discussed with a provider.

What cannot be checked with cloud computing is to see who has access from the provider side. The provider determines which employees have access; however they do not manage access control. Anyone with the login data could access the cloud of an organization and access all their data.

1.6.2 Privacy Issues

Some security issues tend to partially contain privacy issues. This makes sense because they are related to each other. Privacy is in some way determined by how security is handled; therefore it

is not useful to redefine these issues completely. The privacy issues exist because (partial) the infrastructure moves to a provider. Personal data and possibly critical data of organizations move around in the cloud. Because it is out of the viewing range of the organization it is risky as they cannot see who is using the cloud. They must thrust the provider that access is managed and only accessible to authorized personnel.

Another point is about how the cloud is managed, it is important that not everybody has the same rights in the cloud and can see all information. Top management needs distinct information than a simple employee. Besides that, they need other information provision, it would be risky if any employee is be able to see the organizations critical information as this could then easily be leaked to the outside world.

1.7 Identifying Cloud Computing Risks

The previous chapter has provided a good overview of what cloud computing is en what it entails. Some issues and risks have been mentioned there, but this chapter will go more into detail about the risks that cloud computing brings along as stated in previous research. The risks can be divided into two areas; these areas are privacy related risks and (data) security related risks. We will try to give real life examples in relation to the risks to get a clear vision of what impact these risks could have. From a selection of papers we found a very broad paper concerning risks. It shows also to be a good and popular paper with relatively a lot of citations, compared with other papers in that area. Other papers also support statements made in this large cloud risk paper. We can see this paper as a sort of summary of different risk discussing papers.

1.7.1 Privacy and confidentiality risk

A research by [8] for the world privacy forum came up with a whole list of findings in the area of cloud computing. In the next sections, discussion of some of the risks from these researches has been discussed more in detail. It covers most of general risks that cloud computing brings regarding privacy and confidentiality.

The users and clients of cloud computing are dependent on their cloud provider when it comes to their privacy or confidentiality. The provider of the cloud computing services determines what policies are held. Imagine that these providers also have the ability to make changes in their policies. It could completely change the privacy for clients. (For example when the data inserted by the cloud users is protected in the preliminary made up policy being used). Changing policies which will allow insight in this data for third parties could be a serious risk depending on the importance of data that is being used [8]. Another example is that cloud providers could extract information from different organizations in the cloud. They could visualize information that could be by any means revealing. It could also detect information that is commercially valuable for them. What stays important is that most cloud users (clients) are usually not aware of the complete policy and thus do not know very well what risks they are exposed to when entering their data into the cloud [11].

This brings us to the next point where the problem lies in that cloud users share their information with the cloud provider. On itself this is not the problem, but there could (and are) laws in some specific cases that state that certain information is not to be shared with third parties . In this case the third party would be the cloud provider. There are a lot of examples to think about; privacy laws containing specific rules about sharing a client's personal information, such as phone number and address [12]. When an organization uses cloud computing and they put the clients

information in the software that is hosted in the cloud, they are actually sharing the clients information with a third party. These laws and regulations will decline the effect of using cloud computing. Organizations will still need software running on their own servers in order to keep the information which is legally bounded to a certain set of rules.

When there are no laws about sharing certain information with third parties such as cloud computing providers, another problem arises. Sensitive information shared in the cloud might get controlled by weak privacy protection. When this data is stored in your own datacenters you can determine how you want to protect this data and also you are the only organization that is able to access this data. You can choose when and whether or not you want to share this information with certain organization. When all this information is in the cloud, they decide upon over the data privacy. Other organizations could extract the information from the cloud provider more easily then when this data would be stored in your own datacenters. An example is a DNA database. This database could store peoples DNA in order to find a certain cure for a disease. When this database is stored on an own datacenter a hospital or research institute can determine whether they want to share it with a police department for example. The police department could be looking for a fugitive and a DNA database would be handy. Though, when the research institute does not want to provide the database information because they promised their costumers confidentiality, the police department would not get access. On the other side, if this information would be in a cloud, and the cloud provider is not aware of the importance of the data and the confidentiality as promised by the research institute, they would provide this information more easily to a police department.

Earlier we discussed the law concerning privacy and personal information. The laws differ in countries, so important is to know where the data stays in the cloud. In essence the information in

the cloud is stored on a machine that is provided by a certain organization. The laws that apply for the information on this machine depend on the location where it is stored. So for example when you have a service contract with your cloud provider, but your data is (partially) stored in another country with other laws, there are different regulations concerning privacy in this case. Authorities could pressure the cloud provider more easily into handing over the information in the cloud. When the data would always be in the same location (country) this problem would not exist.

The different locations of data storage bring another problem to mind. Different locations (countries) provide different regulations. When a cloud provider moves the data of a user along different countries that are in the cloud, the legislation of the data also changes. This means again that it is difficult to guarantee a certain degree of privacy about the data. For example when a client enters the cloud with their data and with the help of a service contract determines the privacy. This service contract is then bounded to the legislation of that particular country. When the cloud is rearranging their data and the clients data gets moved to another country with other jurisdiction you could get the same problems as described before. Local authorities could pressure the cloud providers in other jurisdictions to provide the information of the cloud.

We have spoken a lot about the law in the previous parts of this chapter. Now we will take a look into the laws themselves. Even though that as an organization you can have a good service contract with a cloud provider there are laws that still override these contracts. Cloud computing provides services for all kind of organizations and people, so it is inevitable that there will be transfer of information concerning such ‘crimes’. The law ‘against terrorism’ in most countries then obligates the cloud provider to pass this information to the authorities in order to prevent

terrorism in this case. The user records could be obligated to be accessible for authorities when they assume there is critical information stored in the cloud.

To continue with the laws around privacy we must stand still with the changing technology. The laws do not change as rapidly as the technologies do. It is commonly known that changes in the law are made very slow. This will result in grey areas with for example cloud computing. Does data needs to be publicly accessible for authorities or not? Such questions provide a certain amount of risk. Governments and authorities could pressure cloud providers to provide cloud information because there is nothing concrete stated in the law about this matter.

Cloud providers should be very familiar with the current laws and regulations in the countries where they provide cloud computing. The privacy and confidentiality risks need to be mentioned in policies and contracts. This would result for user in making more accurate decision about where they will store critical or confidential information

1.7.2 Security risks

Research has been done in the area of cloud computing concerning more technical risks. There are several risks to be found in this area, but we will only discuss the most relevant and important ones for this thesis' subject. These forms of risk have to do with hacking (technical), or attacks from people with malicious intends.

1.7.3 Attacks

The web in general is haunted by attacks on XML signatures. XML is a web based language and as cloud computing could also be web based, they are exposed to this problem. These forms of hack are usually used to obtain data without having the rights to access them. A rightful user does a request for a certain piece of data or information, and the hacker intercepts this request. He then

uses the `sign` of the rightful user in order to obtain the data he wants. He sends his request to the cloud, and because the cloud recognizes him as a valid user it responds with the requested data. This is a dangerous risk because the hacker can act as if he is a legitimate user of the cloud.

An example would be a credit card company that is using the cloud. When an employee of the bank requests for a client's personal data, this request is send to the cloud. The hacker intercepts this request and uses the ´sign´ of this employee. The employee verifies himself by logging into the system with a password and username combination. This provides the sign for that particular employee. Now when the hacker intercepts this request, he can act as being the employee. He is then able to change the request for its personal use. Client's personal data could be changed into client's credit card numbers. The results would speak for itself.

As we started to explain a cloud can be web based. The cloud is a kind of remote server just like we know them now [2]. We connect the client PC´s laptops PDA, etc. to the cloud and use it for input and output. Also important is that it verifies 'us' as users. It gives us thus authorization to access certain areas in the cloud. All these devices themselves do not connect to the cloud; in essence it is the browser on these machines that establishes the actual connections. Most users have Internet Explorer, Firefox or Chrome installed and use this to connect to the cloud. This link is another security risk and needs protection. Browser security is something that depends on the provider of the browsers.

The browsers are used to navigate to the cloud, but also to navigate to other websites. These browsers have to read scripts that are used on the websites. It is important that browsers can detect the difference between malicious scripts that could be made for controlling browser

information. For this issue there is help of a firewall. This security measure is depended on the browser used.

Besides risk in the access tools for the cloud, the cloud itself is also exposed to risks. These risks must be coped with by the provider of cloud services. The first risk described for the cloud is called an Injection attack. These forms of attacks try to implement an own coded malware applications into the existing cloud. The goals of this malware could differ from obtaining only data to completely control applications in the cloud. The software is brought into the cloud, and the cloud is fooled to believe that that software is provided by the cloud user. Whenever cloud users then use the systems, they will be redirected to the malware instead of their real software.

Another important form of risk for a cloud is a so called flooding attack. In general, flooding attacks are to be seen as a huge amount of requests for a service. A "hacker" sends many request to a server which the cloud hosts. These entire requests are actually not genuine and have the goal to get the cloud offline. It tries to make so many requests for a particular service that the server cannot cope with the amount of request so it goes down.

Flooding attacks cause both direct and indirect denial of service. Logically when a cloud finds a lot of requests for a particular server, it accounts additional computing power to that service in order to handle all the requests. This is the general idea of cloud computing. However in the real situation, this would only be in advantage of a "hacker". The hacker now only needs to focus his flooding attack on a single server in order for the cloud to account all the computing power to that service. This is the so called direct denial of service because the hacker focuses on a service and wants to get that particular service down.

On the other side there is indirect denial of service. This then affects other services when an attacker means to hack a particular service down in the direct denial of service. These effects depend on the computing power the hacker has access to. If he tries to cause downtime for a particular service (which is hosted on a server) it could cause downtime for other services too. The servers account all their computing power to all the requests that are being made for one specific service, and thus this causes that there is no rest of computing power to access other applications in the cloud on that particular server. Though it depends on the infrastructure of the cloud, how bad the side effects are. For example the cloud could export the service to another server when it notices that a particular server is not able anymore to cope with all the requests. This will cause even more downtime on other services than before.

1.8 Organization of Thesis

The structure of the thesis is illustrated in Figure 1.1. In more detail, **Chapter 1** provides an overview of Cloud Computing" and the terminologies associated with it .Chapter 1 explains the technology in a elaborated way, its implementation details, benefits of the technology to the IT industry and the pitfalls related with it. All the security concerns that are associated with the Cloud Computing are discussed in Chapter 1.

Furthermore **Chapter 2** elaborates the term "Erasure Coding" which is actually the backbone the proposed approach. It also explains the three important erasure coding schemes (Shamir Secret Algorithm, Reed Solomon Coding and Fountain Codes) that we have taken into consideration in our work.

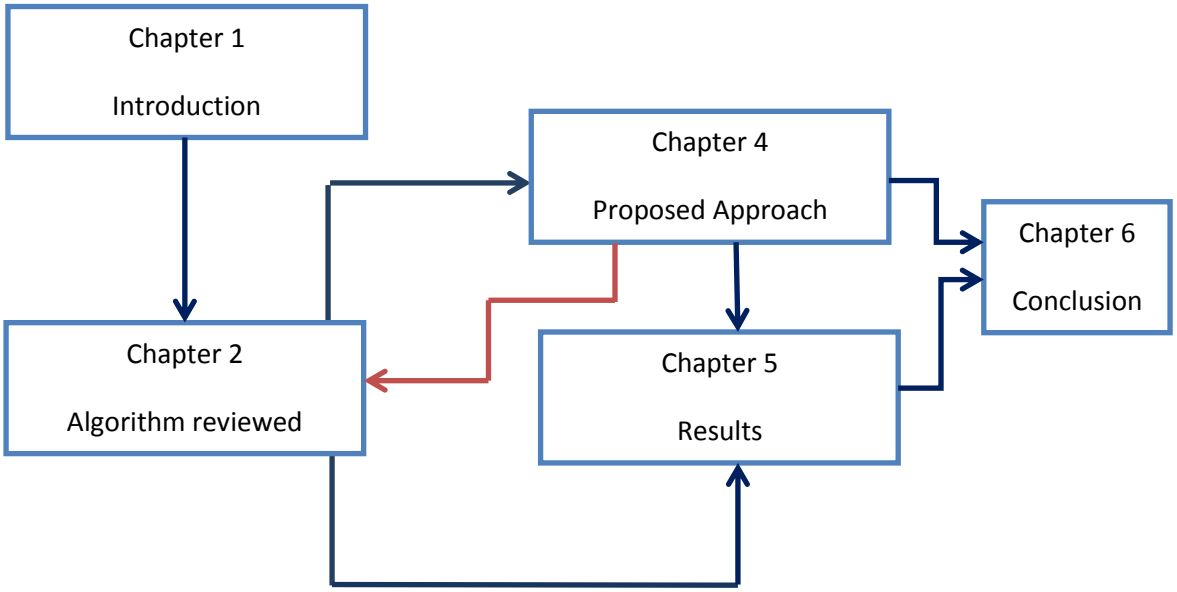

Figure 1.2: Thesis organization roadmap

Chapter 3 describes the proposed approach that is being used for implementing the security in the Cloud architecture. This Chapter explains the security feature which should be adopted at the client side and the server side with its different aspects.

Chapter 4 shows the results that are obtained by implementing the proposed approach and by making certain observations. This chapter justifies the proposed approach in contrast to existing algorithms.

Finally **Chapter 5** concludes the thesis by discussing the overall contribution of the research in the context of related work in the area. In addition, this chapter also discusses the limitations of the approach and points to future research directions.

CHAPTER 2

ALGORITHMS REVIEWED

- **ERASURE CODING**
- **REED SOLOMON CODING**
- **FOUNTAIN CODES**
- **SHAMIR SECRET CODING**
- **HMAC**

In this Chapter, a brief description of the algorithms that are being used on in the research work has been described. This chapter will contain a detailed study of the 3 famous erasure coded techniques (RS code, Fountain Codes, Shamir Secret Scheme) and HMAC code which is used for token generation (see section 3.1)

2.1 Erasure Coding

Storage architecture is always a strong parameter between competing priorities. Performance is not the most critical component, the objects are large, and users will not tolerate a minute latency for an object. Local caching will provide better performance if necessary for repeated viewing.

Secure archive data systems require high data reliability for three reasons:

1. When data is sensitive and must be encrypted, and the loss of one bit of data will render an object unreadable.
2. When data is compressed, the loss of one bit of data may render an object unreadable.
3. Keeping data for a long time increases the risk of corruption, and decreases the ability to reconstruct data from other sources.

Recent empirical studies have established two things about data stored to magnetic disk RAID[13][14][15] arrays that suggest they aren't as reliable as once believed, and may lead to the use of RAID alternatives.

First, drives fail much more frequently than previously thought perhaps up to 1,500 times more frequently than vendor mean time to failure statistics would suggest. By extension,

drive failure frequency places RAID arrays at high risk of multiple concurrent drive failures that can compromise the disk that RAID creates.

Secondly, many studies have confirmed that bit errors, bit rot and silent corruption events problems that can introduce errors at the time data is written to magnetic disk or, alternatively, to the degradation of bit integrity over time pose a much greater challenge to data integrity in RAID arrays than previously thought. Corrupted bits can be limited in their impact to an individual file, or if they occur in the wrong place, they can take down a full RAID set.

RAID used to be the accepted norm for ensuring resilience and accessibility of stored data; however RAID simply does not scale to the extent required by today's data storage needs. As data volumes grow, RAID becomes a nightmare to manage, and introduces tremendous administrative complexity. Furthermore, when using high capacity disk drives, rebuilds take much longer, increasing the probability of multiple failures, which results in an unacceptably high probability of data loss. Double-parity does not change this fact, it simply delays the problem.

Replication, on the other hand, does a very good job, providing resilience and reliability even at tremendous scale. The catch is that, at very large capacities, keeping 2, 3 or even more copies becomes cost prohibitive.

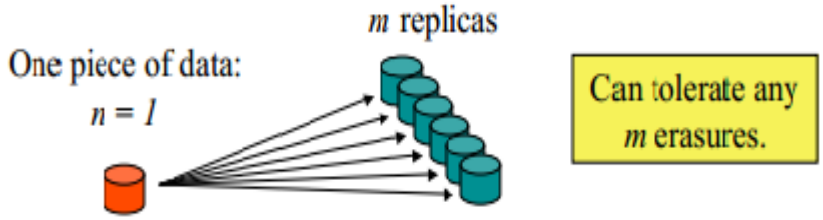

Figure 2.1 : Replication

Simple Parity is also one of the techniques that is used significantly for retrieving the data in case of any server failure. The problem with this technique is that it could be only used in case of any 1 server failure. Multiple failures shall not be tolerated.

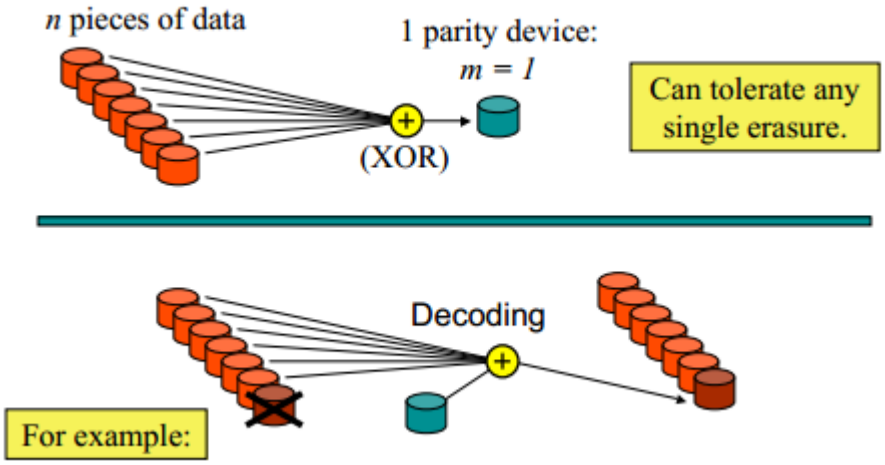

Figure 2.2: Simple Parity Example

Erasure coding was originally conceived as a means to protect data transmitted over unreliable channels or networks. Erasure codes expand data, making data internally redundant so it can be reconstructed from the remaining portion, even if part of the transmitted data is lost. Erasure coding technology addresses both the issues of concurrent multiple failures and the cost concerns associated with large scale deployments.

Erasure coding (EC) is a method of data protection in which data is broken into fragments, expanded and encoded with redundant data pieces and stored across a set of different locations, such as disks, storage nodes or geographic locations.

Erasure coding creates a mathematical function to describe a set of numbers so they can be checked for accuracy and recovered if any of them is lost. Referred to as polynomial interpolation or oversampling, this is the key concept behind erasure codes.

In mathematical terms, the protection offered by erasure coding can be represented in simple form by the following equation: $n = k + m$. The variable "k" is the original amount of data or symbols. The variable "m" stands for the extra or redundant symbols that are added to provide protection from failures. The variable "n" is the total number of symbols created after the erasure coding process.

For instance, in a 10 of 16 configurations, or EC 10/16, six extra symbols (m) would be added to the 10 base symbols (k). The 16 data fragments (n) would be spread across 16 drives, nodes or geographic locations. The original file could be reconstructed from 10 verified fragments.

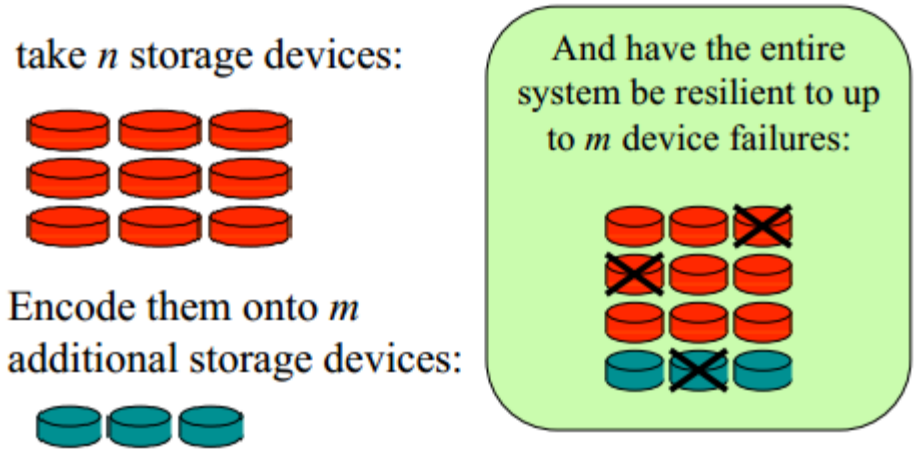

Figure 2.3: Erasure Coding Mechanism

Erasure codes, also known as forward error correction (FEC) codes, were developed more than 50 years ago. Different types have emerged since that time. In one of the earliest and most common types, Reed-Solomon, the data can be reconstructed using any combination of "k" symbols, or pieces of data, even if "m" symbols are lost or unavailable. For example, in EC 10/16, six drives, nodes or geographic locations could be lost or unavailable, and the original file would still be recoverable.

Erasure coding can be useful with large quantities of data and any applications or systems that need to tolerate failures, such as disk array systems, data grids, distributed storage applications, object stores and archival storage. One common current use case for erasure coding is object-based cloud storage [16].

2.2 Reed Solomon Coding

In coding theory Reed Solomon codes are non-binary cyclic error-correcting codes invented by Irving S. Reed and Gustave Solomon. They described a systematic way of building codes that could detect and correct multiple random symbol errors. Reed Solomon forward error correcting codes have become common place in modern digital communications. In actual Reed and Solomons work was based on an area of mathematics invented by French mathematician Evariste Galois in the 1830's. It pains some mathematicians to find that the field of number theory, one of the more esoteric areas of mathematics which Galois helped found, has proved so useful.

Reed Solomon codes work by adding extra information (redundancy) to the original data. The encoded data can then be stored or transmitted. When the encoded data is recovered it may have errors introduced, for instance by scratches on the CD, imperfections on a hard disk surface or radio frequency interference with mobile phone reception. The added redundancy allows a decoder (with certain restrictions) to detect which parts of the received data are corrupted, and correct them. The number of errors the code can correct depends on the amount of redundancy added.

2.2.1 Properties of Reed-Solomon Codes

The topic of error correcting codes is extensive, and most texts treat all codes equally whether they are easily implemented or not. Reed-Solomon codes have certain properties which make them useful in the real world.

RS codes are a *systematic linear block code*. It's a block code because the code is put together by splitting the original message in to fixed length blocks. Each block is further sub divided into m-bit symbols. Each symbol is a fixed width, usually 3 to 8 bits wide.

The linear nature of the codes ensures that in practice every possible m-bit word is a valid symbol. For instance with an 8-bit code all possible 8 bit words are valid for encoding, and you don't have to worry about what data (whether it's binary, ASCII etc.) you are transmitting. Systematic means that the encoded data consists of the original data with the extra 'parity' symbols appended to it.

An RS code is partially specified as an RS (n,k) with m-bit symbols. For instance the DVB code is RS (204,188) using 8-bit symbols. The n refers to the number of encoded symbols in a block, whilst k refers to the number of original message symbols. The difference n-k (usually called 2t) is the number of parity symbols have been appended to make the encoded block. By adding *t* check symbols to the data, an RS code can detect any combination of up to *t* erroneous symbols, or correct up to $\lfloor t/2 \rfloor$ symbols. As an erasure code [section 3.1], it can correct up to *t* known erasures, or it can detect and correct combinations of errors and erasures.

2.2.2 Galois Fields

A finite field Fq is a field F which has a finite number of elements and q is the order of the field. This finite field is often called a Galois field, after the French mathematician Évariste Galois (1811 – 1832) and is denoted GF (q) .

In this thesis only binary field GF (2) and its extension fields GF (2^m) where $m \in \{2, 3, 4, \ldots\}$ is considered. .

The following is always valid for all numbers in a binary Galois field (Blahut,1983):

• fields contain 0 or 1.

• adding two numbers gives one number in the set.

• subtracting two numbers gives one number in the set.

• multiplying one number gives one number in the set.

• dividing one number by 1, as division by 0 is not allowed, gives one numbering the set.

• The distributive law, (a +b) c =ac+bc , holds for all elements in the field.

2.2.3 Problem Specification

Let there be n storage devices, D_1, D2, D3,…Dn, each of which holds K bytes. These are called the *"Data Devices".* Let there be m more storage devices C_1,C_2,…,Cm , each of which also holds k bytes. These are called the *"Checksum Devices".* The contents of the data devices are used to calculate the contents of checksum device. The aim is that if any failure occurs while computing devices contents, then it must be possible to reconstruct those contents from the non-failed devices.

2.2.4 Introduction

This technique is different from others as its main motive is to achieve high bandwidth input and output by distributing data among multiple storage devices. It also uses one to many error correcting devices for failure recovery.

The definition of RAID comes with "Redundant Arrays of Inexpensive Disks (*RAID*)" which means small batteries, low-cost disks bind-up with the high storage capacity, bandwidth and reliability all at a very reasonable price. Therefore, RAID has been used to create layouts of multicomputer and network file systems with high reliability and bandwidth and to sketch fast distributed check pointing systems. These all are termed as "RAID-like" systems [13, 14, 15].

The above problem is central to all RAID-like systems. When storage is divided among n devices, then the chances of device failure becomes more. The clear-cut definition of this is, if

the mean time before failure of one device is F, then the mean time to failure of a system of n devices is F/n. Thus in such systems, fault-tolerance must be taken into picture.

For small values of n and reasonably reliable devices, one checksum device is often sufficient for fault-tolerance. This all defines the "RAID Level 5" configuration, and the coding technique is called *"*n *+1-parity."* With n +1-parity, the i-th byte of the checksum device is measured to be the bitwise exclusive-or (XOR) of the i-th byte of each data device. If failure in any one of the n+ 1 device occurs, then it can be reconstructed as the XOR of the remaining n devices. N+1-parity is pretty-good because of its simplicity. It needs one extra storage device, and one extra write operation per write to any single device. Its major disadvantage is that if many (more than one) simultaneous failure occurs, then it cannot able to recover those devices.

Therefore, everything depends on the number of devices. If n increases, the ability to tolerate multiple failures becomes important. Many techniques have been build-up for this [22, 23], the concentration being small values of m. The most generic technique for tolerating m simultaneous failures with exactly m checksum devices is a technique based on Reed-Solomon coding. Almost all papers on RAID-like systems have noted this fact. However, the technique itself is difficult to come by.

2.2.5 General Strategy

Conventionally, our failure model is that of an *erasure*. When a device fails, it shuts down, and the system recognizes this process. This is as opposed to an *error*, in which a device failure is justified by storing and retrieving incorrect values that can only be understand by sort of embedded coding [24, 25].

For computing the contents of each checksum device Ci, a function Fi is applied to all the data devices. Figure 2.4 gives an example configuration using this technique (which we henceforth

call *"RS-Raid"*) for n=8 and m=2. The contents of checksum devices C1 and C2 are calculated from functions F1 and F2 respectively.

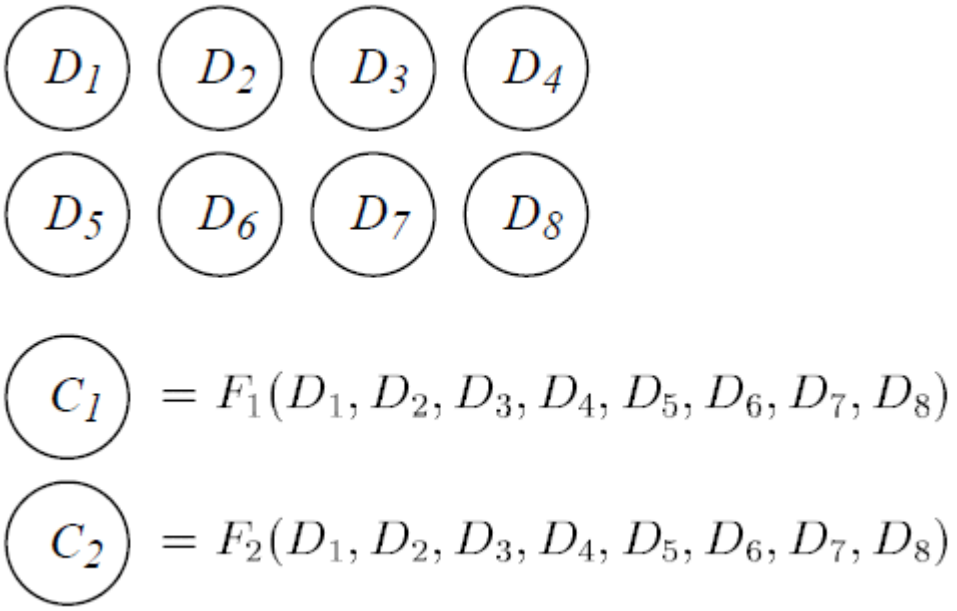

Figure 2.4: Providing two suit Fault Tolerance with two checksum devices

The RS-Raid coding method divides each storage device into *words*. Each word is w bits long, w being chosen by the programmer (subject to some constraints).

Thus, below calculation defines the words present in each storage devices.

$$l = (k \text{ bytes}) \left(\frac{8 \text{ bits}}{\text{byte}}\right) \left(\frac{1 \text{ word}}{w \text{ bits}}\right) = \frac{8k}{w}$$

As Figure 2.5 represents a picture where coding functions Fi, operates on a word-by-word basis. In this, x_{ij} represents the j-th word of device X_i.

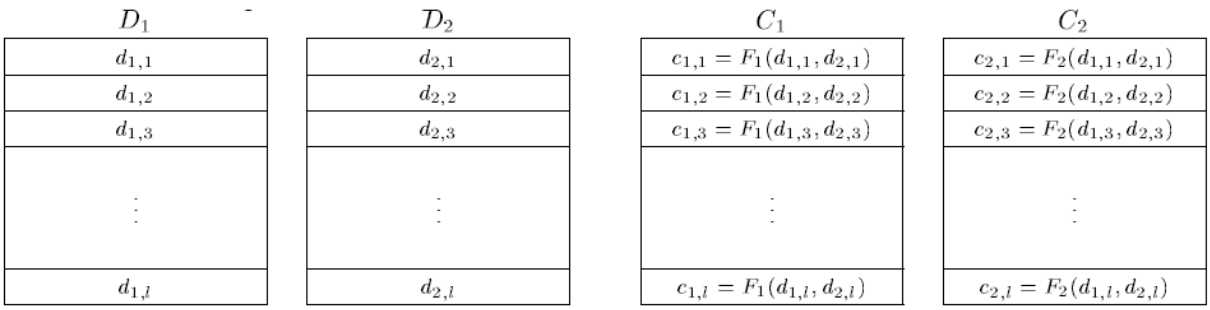

Figure 2.5: Breaking the storage devices into words(n=2, m=2, l=8k/w)

Below assumptions are taken to make it simpler:-

1. Each device contains just one word and drop the extra subscript. Thus it is seen that our problem as consisting of n data words d_1, d2,…..d_n and m checksum words c_1,c_2,…c_m which are calculated from the data words in such a way that the loss of any m words can be acceptable.

To calculate a checksum word c_i for the checksum device C_i, apply function Fi to the data words:

$$c_i = F_i(d_1, d_2, \ldots, d_n).$$

If any modifications to the data word(d_j), on device Dj from dj to d'j found, then each checksum word ci is recalculated by applying a function $G_{i,j}$ such that

$$c_i' = G_{i,j}(d_j, d_j', c_i).$$

Below procedure helps in rebuilding the systems when up to m devices gets failed.

1. For every failed data device D_j we create a function to bring back the words in D_j from the words in the non-failed (correct) devices. When this process gets completed, recalculate any failed checksum devices C_i with F_i.

For example, suppose m=1. Using above study, n +1-parity can also be explained. There is one checksum device C1, and one bit for the word storage (w=1). For calculating each checksum word c1, take the parity (XOR) of the data words:

$$c_1 = F_1(d_1, \ldots, d_n) = d_1 \oplus d_2 \oplus \ldots \oplus d_n.$$

If even a single change on the words on data device D_j occurs, i.e. changes from d_j to d'_j, then c1 is re-evaluated from the parity of its old value and the two data words:

$$c'_1 = G_{1,j}(d_j, d'_j, c_1) = c_1 \oplus d_j \oplus d'_j.$$

If a device D_j failure occurs, then each word may be rebuild as the parity of the corresponding words on the remaining devices:

$$d_j = d_1 \oplus \ldots \oplus d_{j-1} \oplus d_{j+1} \oplus \ldots \oplus d_n \oplus c_1.$$

In such cases, the system recovers quickly to any single device failure.

In the end, the problem statement you have is defined as follow:- You are given with n data words d_1,d2,…..d_n all of size w. As defined, you have functions F and G which are used to compute and preserve the data words. In case of any data loss on device failure (up to m devices) occurs, then you can bring back the words.

So, once you are able to rediscover the data words, checksum words can be easily recalculated using the data words and F.

2.2.6 Overview of the RS-Raid Algorithm

Basically, there are three main aspects of the RS-Raid algorithm which are given as below:

1. Using the Vandermonde matrix to compute and retain checksum words.

2. Using Gaussian Elimination to regain from failures.
3. Using Galois Fields to perform arithmetic.

Detailed explanation is given below:

1. Computing and Maintaining Checksum Words

Each function F_i is defined to be the linear combination of data words:

$$c_i = F_i(d_1, d_2, \ldots, d_n) = \sum_{j=1}^{n} d_j f_{i,j}$$

In other words, if data words and checksum words are represented as vectors D and C, and the functions F_i represented as rows of matrix F, then the below equation can be used to define the system state: $FD = C$. Here F is taken as Vandermonde matrix:

$$f_{i,j} = j^{i-1}.$$

Therefore, the above equation becomes:

$$\begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n} \\ f_{2,1} & f_{2,2} & \cdots & f_{2,n} \\ \vdots & \vdots & & \vdots \\ f_{m,1} & f_{m,2} & \cdots & f_{m,n} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} =$$

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 3 & \cdots & n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \cdots & n^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}.$$

If any change in the data words from d_j to d'_j occurs, then checksum word must be transformed as well. Thus, the result can be given by subtracting out some part of the checksum word that relates to d_j, and adding the appropriate amount to $d'j$.

Therefore, $G_{i,j}$ is defined as:

$$c'_i = G_{i,j}(d_j, d'_j, c_i) = c_i + f_{i,j}(d'_j - d_j).$$

Therefore, simple arithmetic is used to compute and maintain checksum words.

2. Recovering From Failures

To explain this concept, the matrix A and the vector E are defined as

$$A = \begin{bmatrix} I \\ F \end{bmatrix} \text{ and } E = \begin{bmatrix} D \\ C \end{bmatrix}.$$

And the equation is (AD=E).

$$\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ 1 & 2 & 3 & \dots & n \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & 2^{m-1} & 3^{m-1} & \dots & n^{m-1} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_n \\ c_1 \\ c_2 \\ \vdots \\ c_m \end{bmatrix}$$

Each of the devices can be seen as the corresponding rows of matrix A and vector E. The device failure is reflected by removing the device rows from A and from E. And finally, the output contains a new matrix A`, and a new vector E` with an equation:-

$$A'D = E'.$$

Let's say, if exactly m devices gets failed and A's an n x n matrix. As matrix F, is defined as Vandermonde matrix. Therefore, every subset of n rows of matrix A is definitely going to be linearly independent. Thus, the matrix A` is non-singular, and the values of D may be computed from A'D =E` using Gaussian Elimination. Hence all data devices can be restored.

So, once you get the values of D, you can easily recalculate the values of any failed device Ci from D.

Note:- If fewer than m devices fail, then the system recovery is in the manner. That is, by choosing any n rows from A' so as to perform the Gaussian Elimination. Thus, the system tolerance power is up to m device failures.

3. Arithmetic over Galois Fields

A major interest of the RS-RAID algorithm is that the domain and range of the calculation are binary words of a fixed length w. Although the above algebra is surely be correct when all the elements are infinite precision real numbers, we must make sure that it is accurate for these fixed size words. A common error while working with these codes is, to perform all arithmetic over the integers modulo 2^w. This *does not work,* as division is not explained for all pairs of elements (for example, (3/2) is undefined modulo 4), rendering the Gaussian Elimination unsolvable in many cases. Instead, addition and multiplication are executed over a *field* with more than n+m elements.

Fields with 2^w elements are called *Galois Fields* (denoted GF (2^w)). This segment describes how to perform addition, subtraction, multiplication, and division effectively over a Galois Field. In this thesis such a description without describing Galois Fields in general is provided. Appendix A contains a more description of Galois Fields, and gives the justification for the arithmetic algorithms in this part.

The elements of GF (2^w) are the integers from zero to 2^w -- 1 .Addition and subtraction of elements of GF (2^w) are quite easy to perform. They are the XOR operation. For example, in $GF(2^4)$:

$$11 + 7 = 1011 \oplus 0111 = 1100 = 12.$$

Here, multiplication and division operations are difficult to perform.

An Example

Let's suppose we have three data devices and three checksum devices, each of which holds one megabyte. Then and n=3 and m=3, select w to be four, since 2^w > n + m ,

Construct F to be a 3*3 Vandermonde matrix, defined over $GF(2^4)$:

$$F = \begin{bmatrix} 1^0 & 2^0 & 3^0 \\ 1^1 & 2^1 & 3^1 \\ 1^2 & 2^2 & 3^2 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 5 \end{bmatrix}$$

Now, calculate each word of each checksum device using FD=C. For example, suppose the first word of D_2 is 13, the first word of $D_3$is 9, and the first word of D_1 is 3. Then use F to calculate the first words of C_1, C2 and C_3 :

$$
\begin{aligned}
C_1 &= (1)(3) \oplus (1)(13) \oplus (1)(9) \\
&= 3 \oplus 13 \oplus 9 \\
&= 0011 \oplus 1101 \oplus 1001 = 0111 = 7 \\
C_2 &= (1)(3) \oplus (2)(13) \oplus (3)(9) \\
&= 3 \oplus 9 \oplus 8 \\
&= 0011 \oplus 1001 \oplus 1000 = 0010 = 2 \\
C_3 &= (1)(3) \oplus (4)(13) \oplus (5)(9) \\
&= 3 \oplus 1 \oplus 11 \\
&= 0011 \oplus 0001 \oplus 1011 = 1001 = 9
\end{aligned}
$$

Suppose someone changes D_2 to be 1.Then D_2 sends the value (1 - 13)= (0001$\oplus$1101) = 12 to each checksum device, which uses this value to recalculate its checksum.

$$
\begin{aligned}
C_1 &= 7 \oplus (1)(12) = 0111 \oplus 1100 = 11 \\
C_2 &= 2 \oplus (2)(12) = 2 \oplus 11 = 0010 \oplus 1011 = 9 \\
C_3 &= 9 \oplus (4)(12) = 9 \oplus 5 = 1001 \oplus 0101 = 12
\end{aligned}
$$

Suppose now that devices D_2 , D_3 and C_3 are lost. Then delete the rows of A and E corresponds to D_1,D_2 and C_3 to get A'D=E'

$$
\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} D = \begin{bmatrix} 3 \\ 11 \\ 9 \end{bmatrix}
$$

By applying Gaussian elimination, 'A' can be inverted to yield the following equation: $D=(A')^{-1}E'$,

$$D = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 1 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 3 \\ 11 \\ 9 \end{bmatrix}$$

from this the result is

$$D_2 = (2)(3) \oplus (3)(11) \oplus (1)(9) = 6 \oplus 14 \oplus 9 = 1$$

$$D_3 = (3)(3) \oplus (2)(11) \oplus (1)(9) = 5 \oplus 5 \oplus 9 = 9$$

And then

$$C_3 = (1)(3) \oplus (4)(1) \oplus (5)(9) = 3 \oplus 4 \oplus 11 = 12$$

And finally the system gets recovered.

2.3 Fountain Codes

In this section Fountain Codes, otherwise known as "rate less codes" are explained. Fountain codes are priceless in the sense that the number of encoded packets that can be created from the source message is potentially endless; and the number of encoded packets created can be dynamically resolved [17]. Fountain codes are standard because they are concurrently near-optimal for each erasure channel. Despite of the statistics of the erasure events on the channel, we can send any number of encoded packets in order for the decoder to regain the source data. The source data can be decoder from any set of K' encoded packets, for K' somewhat larger than K. Fountain codes can also have fabulously small encoding and decoding complexities.

A fountain code is a way to take some data - a file, for example - and transform it into an effectively unlimited number of encoded chunks, such that you can reassemble the original file given any subset of those chunks, as long as you have a little more than the size of the original file. In other words, it lets you create a 'fountain' of encoded data; a receiver can reassemble the file by catching enough 'droplets', regardless of which ones they get and which ones they miss. There are a number of variants, of Fountain Code like an LT or Luby Transform Code, Tornado Codes and Raptor Codes.

2.3.1. The LT code

The LT code preserves the good performance of the random linear fountain code, while radically decreasing the encoding and decoding complexities. You can think of the LT code as a sparse random linear fountain code, with an approximately decoding algorithm which costs to almost equal to nothing.

LT codes generate encoded blocks like this:

1. Pick a random number, d, between 1 and k, the number of blocks in the file. We'll discuss how best to pick this number later.
2. Pick d blocks at random from the file, and combine them together. For our purposes, the XOR operation will work fine.
3. Transmit the combined block, along with information about which blocks it was constructed from.

A lot depends on how we pick the number of blocks to combine together , called the degree distribution - but we'll cover that in more detail shortly. It can be observed from the description that some encoded blocks will end up being composed of just a single source block, while most will be composed of several source blocks.

Another thing that might not be immediately obvious is that while we do have to let the receiver know what blocks we combined together to produce the output block, we don't have to transmit that list explicitly. If the transmitter and receivers agree on a pseudo-random number generator, we can seed that PRNG with a randomly chosen seed, and use that to pick the degree and the set of source blocks. Then, we just send the seed along with the encoded block, and our receiver can use the same procedure to reconstruct the list of source blocks we used.

The decoding procedure is a little but not much more complicated:

1. Reconstruct the list of source blocks that were used to construct this encoded block.
2. For each source block from that list, if we have already decoded it, XOR that block with the encoded block, and remove it from the list of source blocks.

3. If there are at least two source blocks left in the list, add the encoded block to a holding area.
4. If there is only one source block remaining in the list, we have successfully decoded another source block. Add it to the decoded file, and iterate through the holding list, repeating the procedure for any encoded blocks that contain it.

Suppose we receive five encoded blocks, each one byte long, along with information about which source blocks each is constructed from. Fig 2.6 represents the data blocks.

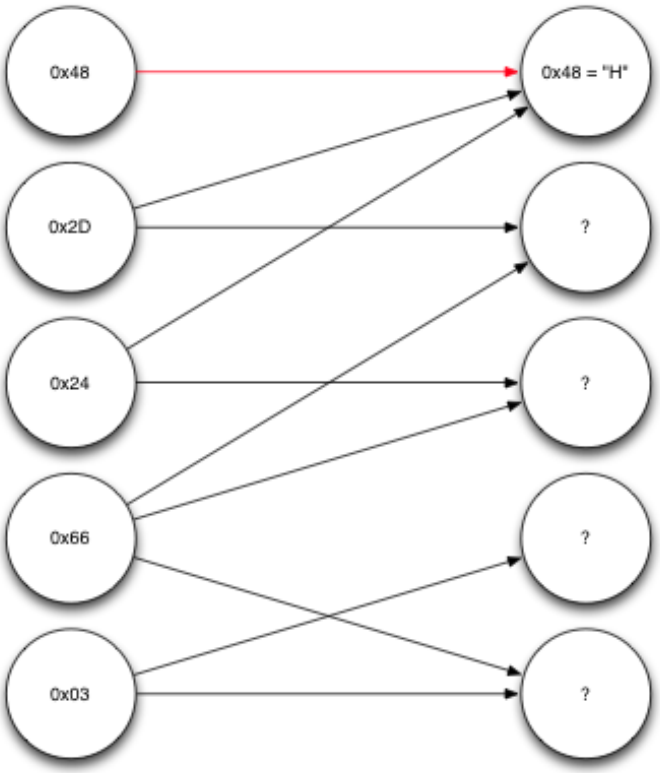

Figure 2.6: Data blocks encoding

Nodes on the left represent encoded blocks we received, and nodes on the right represent source blocks. The first block we received, 0x48 turns out to consist of only one source block - the first source block - so we already know what that block was [21].

Following the arrows pointing to the first source block back, we can see that the second and third encoded blocks only depend on the first source block and one other, and since we now know the first source block, we can XOR them together and thus we can know the third source block also which will give the output as shown in Figure 2.7.

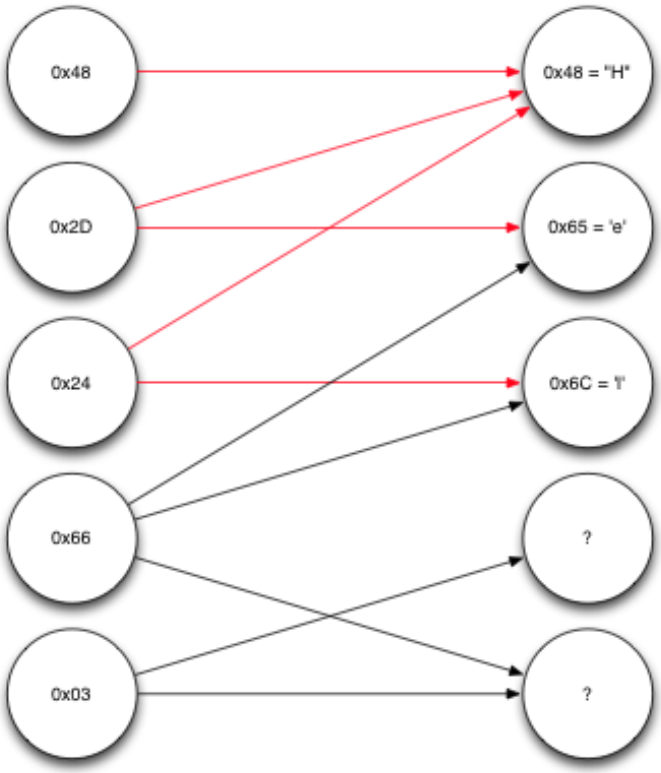

Figure 2.7: Finding the three source block

Repeating the same procedure again, we can see we now know enough to decode the fourth encoded block, which depends on the second and third source blocks, both of which we now

know. XORing them together lets us decode the fifth and final source block, as shown in Figure 2.8.

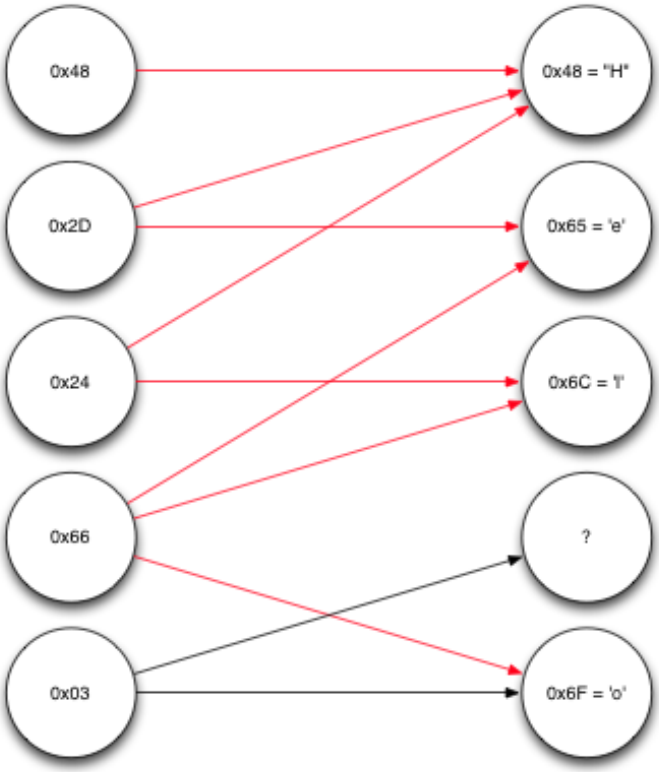

Figure 2.8: Finding the final data block

Finally, we can now decode the last remaining source block, giving us the rest of the message and the final output with the help of pre calculated packets.

We present an example also of the LT codes so as to understand it more clearly. Figure 2.9 shows the problem and its solution.

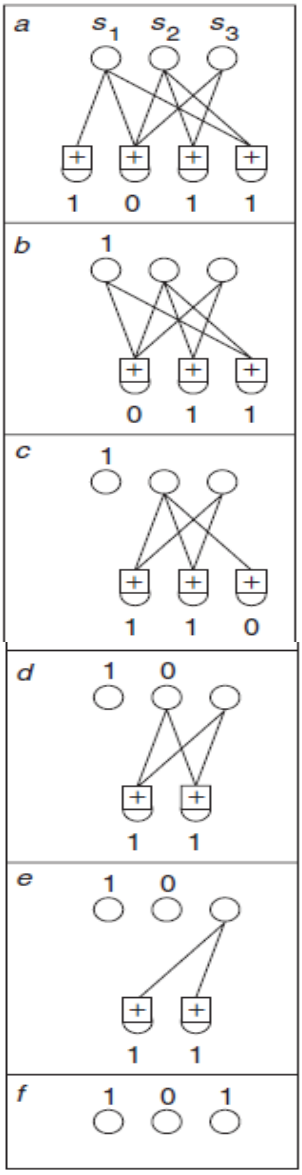

Figure 2.9: Example 2 of LT codes

At the first iteration, the only check node that is connected to a sole source bit is the first check node (panel a).

We set that source bit s_1 therefore (panel b), reject the check node, then add the value of s_1 (1) to the checks to which it is joined (panel c), disjoin s1 from the graph. At the start of the second repetition (panel c), the fourth check node is coupled to a sole source bit, s2. We set s_2 to t_4 (0, in

panel d), and add s_2 to the two checks it is joined to (panel e). Finally, we get that two check nodes are both joined to s_3, and they agree about the value of s_3 (as we would hope!), which is restored in panel f.

Unfortunately, the ideal soliton distribution isn't quite so ideal in practice, as random variations make it likely that there will be source blocks that are never included, or that decoding stalls when it runs out of known blocks. A variation on the ideal soliton distribution, called the robust soliton distribution [20], improves on this, generating more blocks with very few source blocks, and also generating a few blocks that combine all or nearly all of the source blocks, to facilitate decoding the last few source blocks [20].

2.3.2 Raptor codes

You might think that we could not do any better than LT codes: their encoding and decoding costs scale as K loge K, where K is the file size. But raptor codes [19] achieve linear time encoding and decoding by concatenating a weakened LT code with an outer code that patches the gaps in the LT code.

2.4 Shamir Secret Algorithm

Shamir's Secret Sharing is one of the famous algorithms in cryptography. It also known as secret sharing [31, 32], in which a secret is divided into parts, and each participant is given his own part. The division is done in such a manner that some of the parts or all of them are needed in order to reconstruct the secret. Counting on all participants to combine together the secret might be impractical, and therefore sometimes the *threshold scheme* is used where any k of the parts are sufficient to reconstruct the original secret [33].

We can use this algorithm in cloud architecture in way that data will be divided on to a number of servers and after that a subset of servers will be required to construct the complete data.

2.4.1 Mathematical Definition

Formally, our goal is to divide some data D (e.g., the safe combination) into n pieces $D_1, \ldots, D_n$ in such a way that:

1. Knowledge of any k or more D_i pieces makes D easily computable.
2. Knowledge of any $k-1$ or fewer D_i pieces leaves D completely undetermined (in the sense that all its possible values are equally likely).

This scheme is called *(k,n)* threshold scheme. If $k = n$ then all participants are required to reconstruct the secret [33].

2.4.2 Shamir Secret Sharing Scheme

The main idea behind Adi Shamir's threshold scheme is that 2 points are sufficient to define a line, 3 points are sufficient to define a parabola, 4 points to define a cubic curve and so forth. That is, we require n points to define a polynomial of degree n-1.

Suppose we want to use a *(k,n)* threshold scheme to share our secret *S*, without loss of generality assumed to be an element in a finite field F of size 0<k<=n<P where P is a prime number.

Choose at random $k-1$ coefficients $a_1, \cdots, a_{k-1}$ in F, and let $a_0 = S$. Build the polynomial $f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_{k-1} x^{k-1}$. Let us construct any n points out of it, for instance set i=1,2……*n* to retrieve *(i,f(i))*. Every participant is given a point (a pair of input to the polynomial and output). Given any subset of k of these pairs, we can find the coefficients of the polynomial using interpolation and the secret is the constant term a_0.

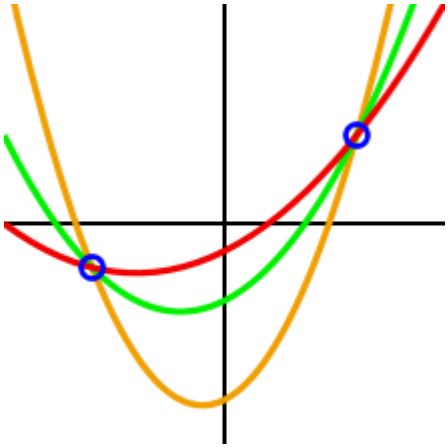

Figure 2.10: One can draw infinite polynomials an infinite number of polynomials of degree 3 through 2 points.3 points are required to define a unique polynomial of degree 2

2.4.3 Example

The example shown below demonstrates the basic idea of Shamir Secret Scheme. Note, however, that calculations in the example are done using integer arithmetic rather than using finite field[29,30] arithmetic. Therefore the example below does not provide perfect secrecy, and is not a true example of Shamir's scheme.

Preparation

Suppose that our secret is 1234 . (*S=1234)*

We wish to divide the secret into 6 parts *(n=6)*, where any subset of 3 parts *(k=3)* is sufficient to reconstruct the secret. At random we obtain two (*k-1*) numbers: 166 and 94.

$$(a_1 = 166; a_2 = 94)$$

Our polynomial to produce secret shares (points) is therefore:

$$f(x) = 1234 + 166x + 94x^2$$

We construct 6 points from the polynomial:

$$(1, 1494); (2, 1942); (3, 2578); (4, 3402); (5, 4414); (6, 5614)$$

We give each participant a different single point (both x and *f(x)*).

Reconstruction

In order to reconstruct the secret any 3 points will be enough.

Let us consider $(x_0, y_0) = (2, 1942); (x_1, y_1) = (4, 3402); (x_2, y_2) = (5, 4414)$.

We will compute Lagrange basis polynomials [27, 28]:

$$\ell_0 = \frac{x - x_1}{x_0 - x_1} \cdot \frac{x - x_2}{x_0 - x_2} = \frac{x - 4}{2 - 4} \cdot \frac{x - 5}{2 - 5} = \frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}$$

$$\ell_1 = \frac{x - x_0}{x_1 - x_0} \cdot \frac{x - x_2}{x_1 - x_2} = \frac{x - 2}{4 - 2} \cdot \frac{x - 5}{4 - 5} = -\frac{1}{2}x^2 + \frac{7}{2}x - 5$$

$$\ell_2 = \frac{x - x_0}{x_2 - x_0} \cdot \frac{x - x_1}{x_2 - x_1} = \frac{x - 2}{5 - 2} \cdot \frac{x - 4}{5 - 4} = \frac{1}{3}x^2 - 2x + \frac{8}{3}$$

Therefore

$$f(x) = \sum_{j=0}^{2} y_j \cdot \ell_j(x)$$

$$= 1942 \cdot \left(\frac{1}{6}x^2 - \frac{3}{2}x + \frac{10}{3}\right) + 3402 \cdot \left(-\frac{1}{2}x^2 + \frac{7}{2}x - 5\right) + 4414 \cdot \left(\frac{1}{3}x^2 - 2x + \frac{8}{3}\right)$$

$$= 1234 + 166x + 94x^2$$

Recall that the secret is the free coefficient, which means that *S=1234*, and we are done.

2.5 Hashed Message Authentication Code

Providing a way to check the integrity of information transmitted over or stored in an unreliable medium is a prime necessity in the world of open computing and communications. Mechanisms that are able to perform such integrity check based on a secret key are usually called "message authentication codes" (MAC). Typically, message authentication codes are used between two parties that share a secret key in order to validate information transmitted between these parties. In this report we present such a MAC technique based on cryptographic hash functions. This mechanism is called HMAC.

HMAC can be used in combination with any of the iterated cryptographic hash function. MD5 and SHA-1 are examples of such hash functions. HMA also uses a secret key for calculation and verification of the message authentication values.

Note: MD5 and SHA-1 are the most widely used cryptographic hash functions MD5 has been recently shown to be vulnerable to collision search attacks [Dobb].However, SHA-1 appears to be a cryptographically stronger function. To this date, MD5 can be considered for use in HMAC for applications where the superior performance of MD5 is critical. In any case, implementers and users need to be aware of possible cryptanalytic developments regarding any of these cryptographic hash functions, and the eventual need to replace the underlying hash function.

2.5.1 Definition of HMAC

The definition of HMAC requires a cryptographic hash function, which we denote by H, and a secret key K. We assume H to be a cryptographic hash function where data is hashed by iterating a basic compression function on blocks of data. We denote by B the byte-length of such blocks (B=64 for all the above mentioned examples of hash functions), and by L the byte-length of hash

outputs (L=16 for MD5, L=20 for SHA-1). The authentication key K can be of any length up to B, the block length of the hash function. Applications that use keys longer than B bytes will first hash the key using H and then use the resultant L byte string as the actual key to HMAC. In any case the minimal recommended length for K is L bytes (as the hash output length).

We define two fixed and different strings ipad and opad as follows (the 'i' and 'o' are mnemonics for inner and outer):

ipad = the byte 0x36 repeated B times

opad = the byte 0x5C repeated B times.

To compute HMAC over the data `text' we perform

H(K XOR opad, H(K XOR ipad, text))

Namely,

(1) Append zeros to the end of K to create a B byte string (e.g., if K is of length 20 bytes and B=64, then K will be appended with 44 zero bytes 0x00)

(2) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with ipad.

(3) Append the stream of data 'text' to the B byte string resulting from step (2).

(4) Apply H to the stream generated in step (3)

(5) XOR (bitwise exclusive-OR) the B byte string computed in step (1) with opad.

(6) Append the H result from step (4) to the B byte string resulting from step (5)

(7) Apply H to the stream generated in step (6) and output the result.

2.5.2 Keys

The key for HMAC can be of any length (keys longer than B bytes are first hashed using H).

However, less than L bytes are strongly discouraged as it would decrease the security strength of the function. Keys longer than L bytes are acceptable but the extra length would not significantly increase the function strength. (A longer key may be advisable if the randomness of the key is considered weak.)Keys need to be chosen at random (or using a cryptographically strong pseudo-random generator seeded with a random seed), and periodically refreshed. (Current attacks do not indicate a specific recommended frequency for key changes as these attacks are practically infeasible. However, periodic key refreshment is a fundamental security practice that helps against potential weaknesses of the function and keys, and limits the damage of an exposed

key.)

2.5.3 Implementation Note

HMAC is defined in such a way that the underlying hash function H can be used with no modification to its code. In particular, it uses the function H with the pre-defined initial value IV (a fixed value specified by each iterative hash function to initialize its compression function). However, if desired, a performance improvement can be achieved at the cost of (possibly) modifying the code of H to support variable IVs.

The idea is that the intermediate results of the compression function on the B-byte blocks (K XOR ipad) and (K XOR opad) can be recomputed only once at the time of generation of the key K, or before its first use. These intermediate results are stored and then used to initialize the IV of H each time that a message needs to be authenticated. This method saves, for each authenticated message, the application of the compression function of H on two B-byte blocks (i.e., on (K XOR ipad) and (K XOR opad)). Such a savings may be significant when

authenticating short streams of data. We stress that the stored intermediate values need to be treated and protected the same as secret keys.

Choosing to implement HMAC in the above way is a decision of the local implementation and has no effect on inter-operability.

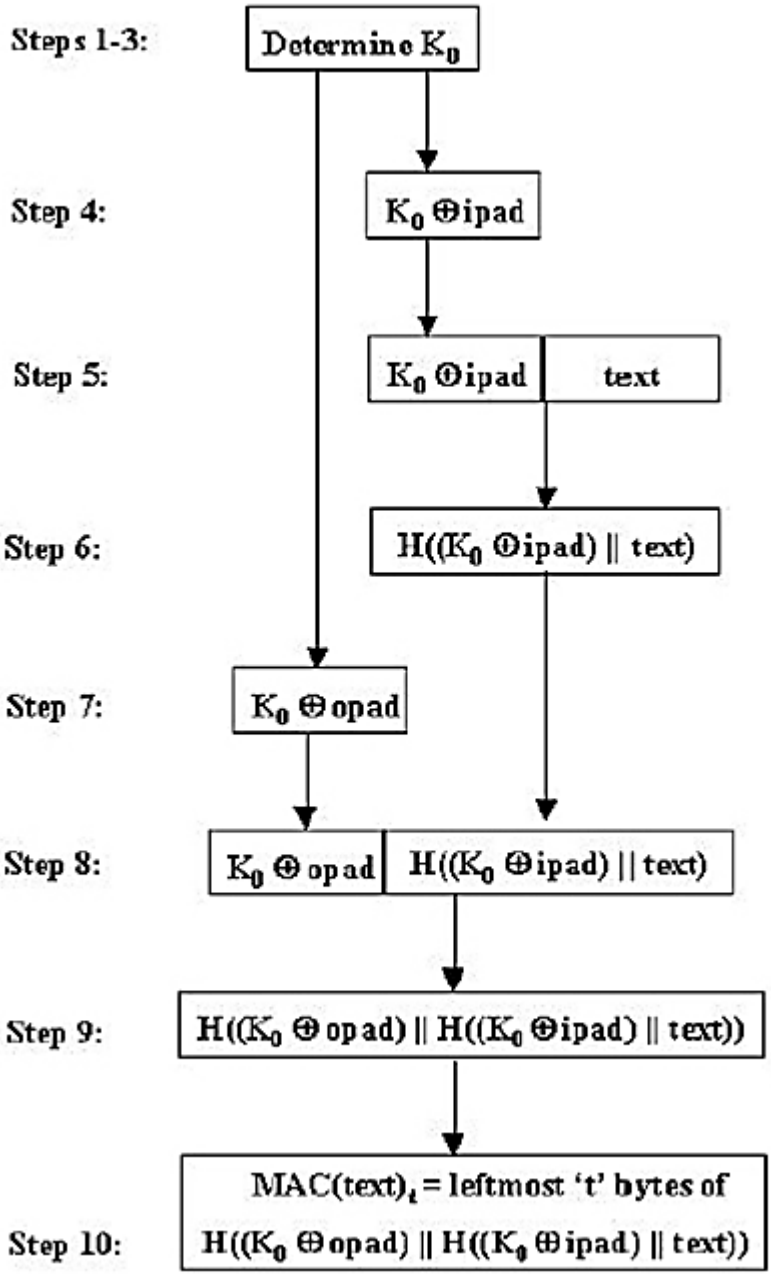

Figure 2.11: Illustrating HMAC generation

2.5.4 Security

The security of the message authentication mechanism presented here depends on cryptographic properties of the hash function H: the resistance to collision finding (limited to the case where the initial value is secret and random, and where the output of the function is not explicitly available to the attacker), and the message authentication property of the compression function of H when applied to single blocks (in HMAC these blocks are partially unknown to an attacker as they contain the result of the inner H computation and, in particular, cannot be fully chosen by the attacker).

As an example, if we consider a hash function like MD5 where the output length equals L=16 bytes (128 bits) the attacker needs to acquire the correct message authentication tags computed (with the same_ secret key K!) on about 2**64 known plaintexts. This would require the processing of at least 2**64 blocks under H, an impossible task in any realistic scenario (for a block length of 64 bytes this would take 250,000 years in a continuous 1Gbps link, and without changing the secret key K during all this time). This attack could become realistic only if serious flaws in the collision behavior of the function H are discovered (e.g. Collisions found after 2**30 messages). Such a discovery would determine the immediate replacement of the function H (the effects of such failure would be far more severe for the traditional uses of H in the context of digital signatures, public key certificates, etc.).

Note: this attack needs to be strongly contrasted with regular collision attacks on cryptographic hash functions where no secret key is involved and where 2**64 off-line parallelizable operations suffice to find collisions. The latter attack is approaching feasibility [VW] while the birthday attack on HMAC is totally impractical. (In the above examples, if one uses a hash function with, say, 160 bit of output then 2**64 should be replaced by 2**80).

CHAPTER 3

PROPOSED ALGORITHM

- **TOKEN GENERATION**
- **DATA DISTRIBUTION ON DISTRIBUTED SERVERS**

In this chapter the proposed algorithm is discussed. The scheme consists of two phases and phase 1 is described below.

3.1 Token Generation

In the first step we propose the use of token generation in order to ensure data integrity to the user. The token computation function we are considering belongs to a family of universal hash function [11], chosen to preserve the homomorphic properties, which can be perfectly integrated with the verification of erasure-coded data. In order to achieve assurance of data storage integrity and data error localization simultaneously, our scheme entirely relies on the pre-computed verification tokens. The main idea is as follows: before file distribution on the servers the user pre-computes a certain set of short verification tokens on individual vector G(j) ($j \in \{1, \ldots, n\}$) where each token covering a random subset of coded data blocks. Later, when the user wants to make sure the storage correctness for the data in the cloud, he challenges the cloud servers with a set of randomly generated block indices and a key so as to create the message code of the blocks. Upon receiving challenge, each cloud server computes a short "signature" over the specified blocks with the help of the received key and returns them to the user. The values of these signatures should match the corresponding tokens pre-computed by the user.

Suppose the user wants to challenge the cloud server's t times to ensure the correctness of data storage. Then, he must pre-compute t verification tokens for each G(j) ($j \in \{1, \ldots, n\}$) with the help of a function f(data) and a key k. User then selects a set of randomly generated block of indices accompanied with a key would be transmitted to the cloud servers to create the message code for the blocks. The servers would do the required computation to compute the message from the key and given block indices and return a small representative "signature" to the user. The

matching of the server side created signature with the user pre computed tokens marks successful data integrity and error localization.

The detailed working of this function is described below.

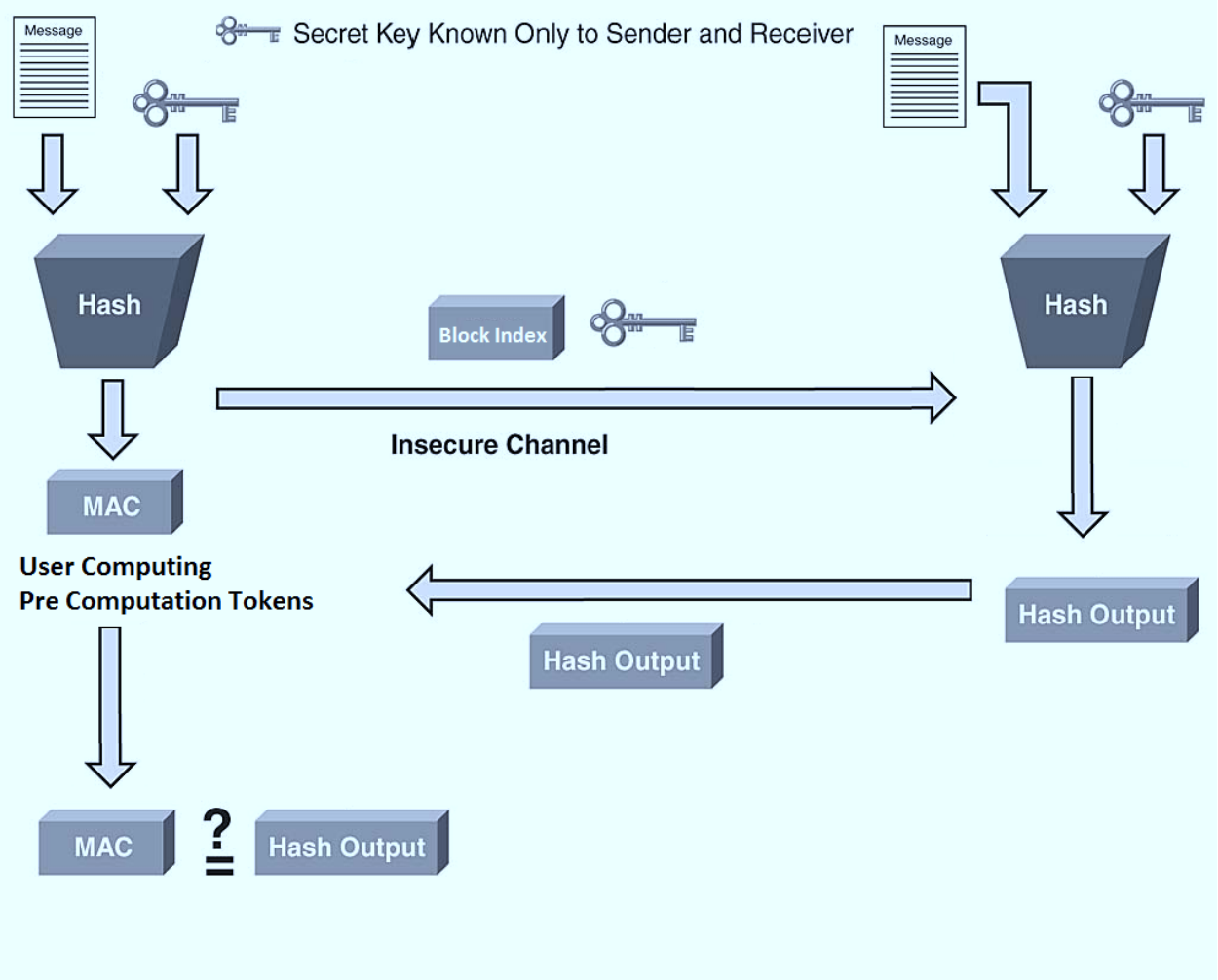

Figure 3.1: Homomorphic token generation process

3.2 Data distribution on distributed servers

This is the second phase of the proposed approach. This section explains the algorithm which can be used to ensure the data integrity and its efficient retrieval in case of any server failure.

3.2.1 Prerequisite of Algorithm

We are assuming that the data will be stored in the distributed architecture on the cloud. So we are considering a main server and some distributed servers where the data will be stored. The below section will discuss the figure's various parts and their working.

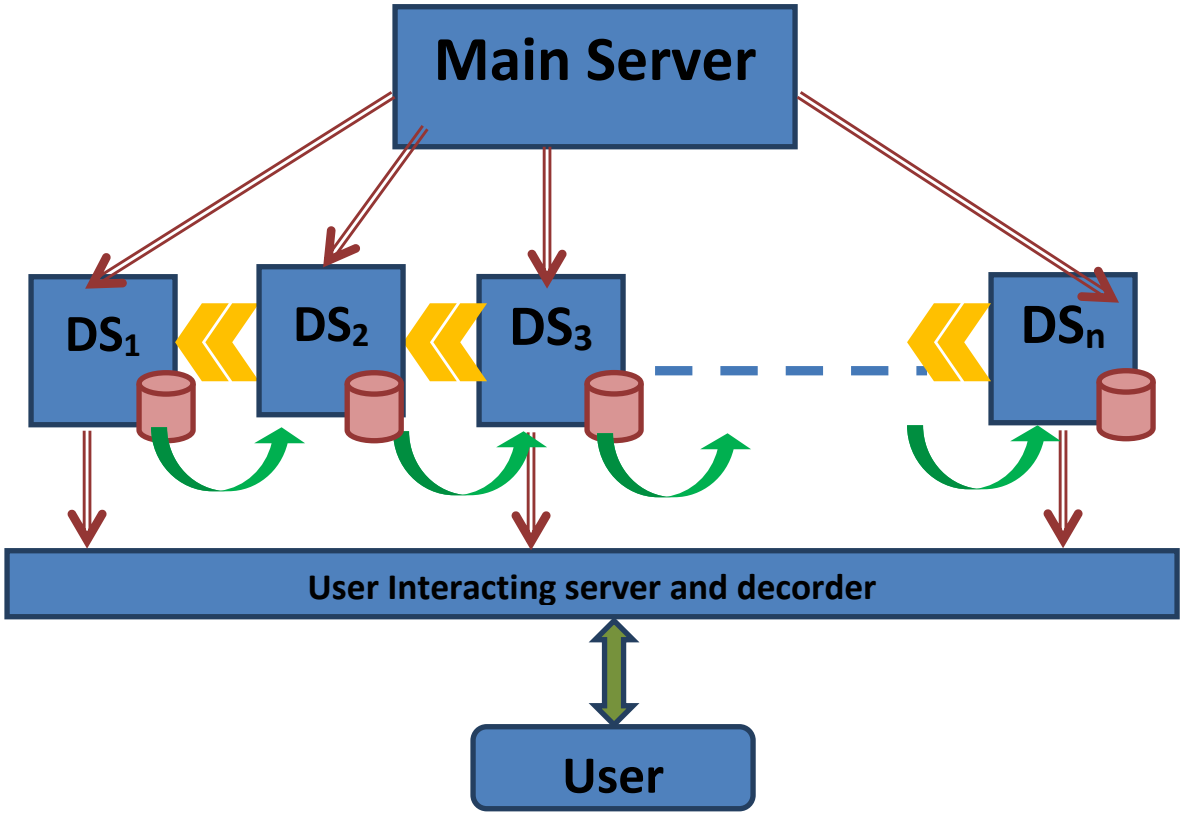

Figure 3.2: Drop Distribution in distributed system

Main Server

Main server is the actual storage device where the data will be stored initially and will be distributed after that. Main server is solely responsible for distributing the preprocessed data to distributed servers and also keeps track of these servers which are alive and can provide the data for its retrieval by the user.

When the data comes to the Main Server, it encodes the data into the various drops which is already explained in the previous sections. It also keeps track of the components using which drop is being made.

Distributed Servers

Distributed servers are denoted by $D_1, D_2 \ldots D_n$ in the Figure 4.3. Each distributed server stores two things-:

1. The drop which is being given by the Main Sever.
2. The information of the components of the drop which is stored on the next consecutive server.

User Interacting server and Decoder

User interface acts as a platform between user and the servers. User will pass its query for retrieval of the data to the servers through the help of User Interacting server. This server also acts as a decoder for the drops that are being received by the distributed servers and passes the decoded data to the user.

User

User could be any individual or any organization who wants to fetch its data from the cloud.

3.2.2 Working of the algorithm

Initially when the data comes to the Main server, it is being encoded in the form of the drops by the Main Server. Main Server made at least as many number of drops as the number of the servers. It also keep record of the components through which each drop is being made.

After the drop creation process, Main Server distributes drop to each server and it also stores additional information of the drop components of the next server. Except the last server every server stores two data structures (drop and the drop component data).

After receiving the drops, each server continuously pings its next server, to check its state, whether active or failed as shown in the figure.

When the user asks for data retrieval, each server passes its drop to the Interacting server and decoder. Decoder decodes the data from the drop and passes the data to the user through the User Interface. But the above is only true in case of no server failure. In Case of any server failure the process is described in the next section.

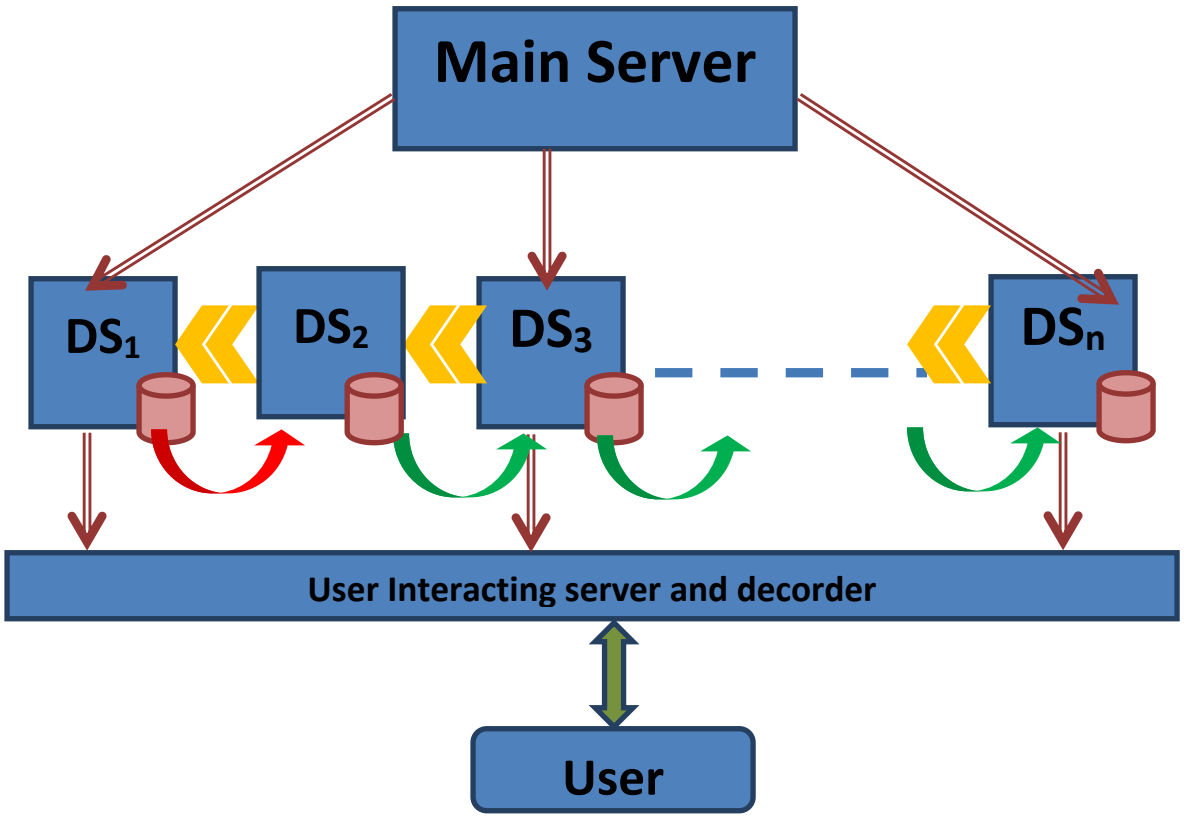

Figure 3.3: Failure of server 2

Figure 3.3 showcases the failure of the second distributed servers. After receiving the drop from the main server, each server pings to its next consecutive server. Any failure in the server will be first observed by the server preceding it. As shown in the figure, at the moment server 2 failed, the failure is first observed by server1 as its ping request fails which is shown by red arrow in the Figure 3.3.

As soon as server 1 realizes that the server succeeding it has failed, it immediately reports to Main server about the same. Server1 also passes the information of the drop components of server2 to the Main server which will help the main server to make new drop.

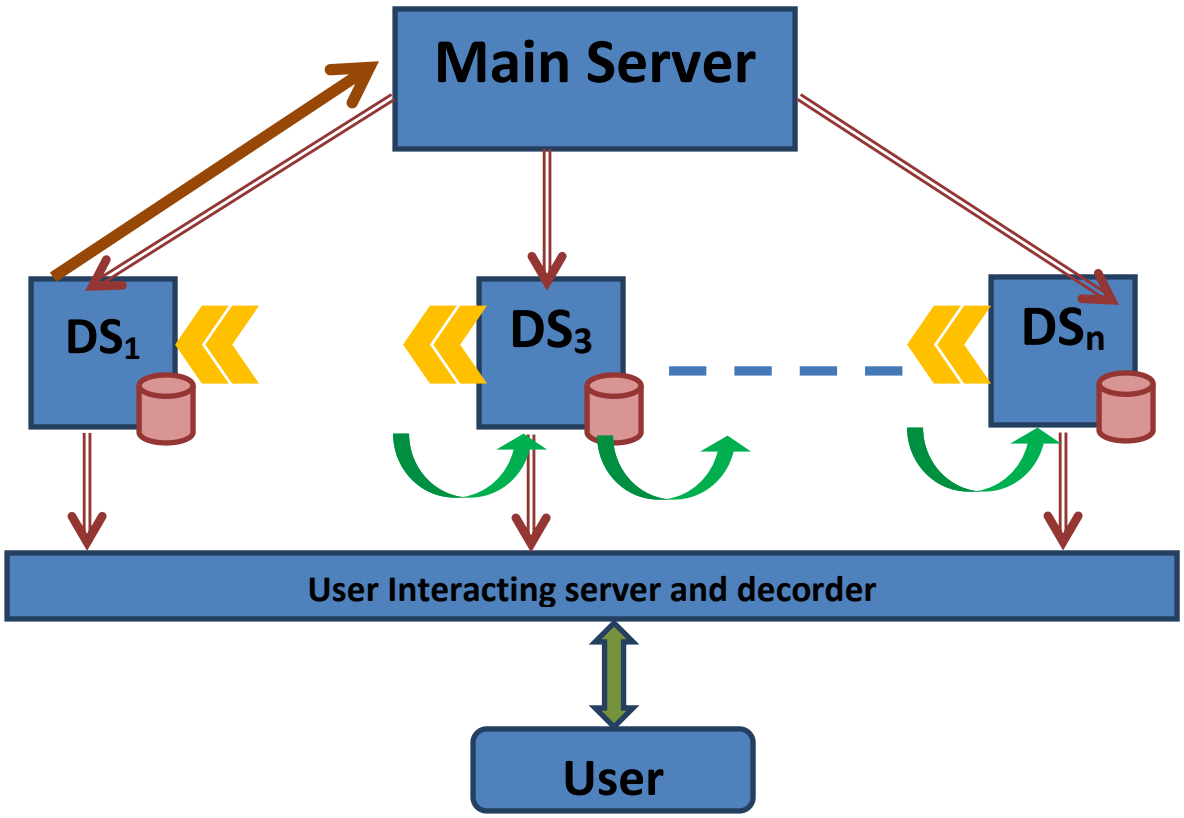

Figure 3.4 : Server reports the server 2 failure to main server

As already discussed in section 2.3, Main server is capable of making infinite number of drops. So, in the final step Main server creates a new drop of the server 2 and passes it to server 1.After getting two drops server1 and all the remaining alive servers provides the necessary drops to the

decoder. Decoder decodes the drops that are received from the alive distributed servers and then decodes the data and finally passes the data to the user.

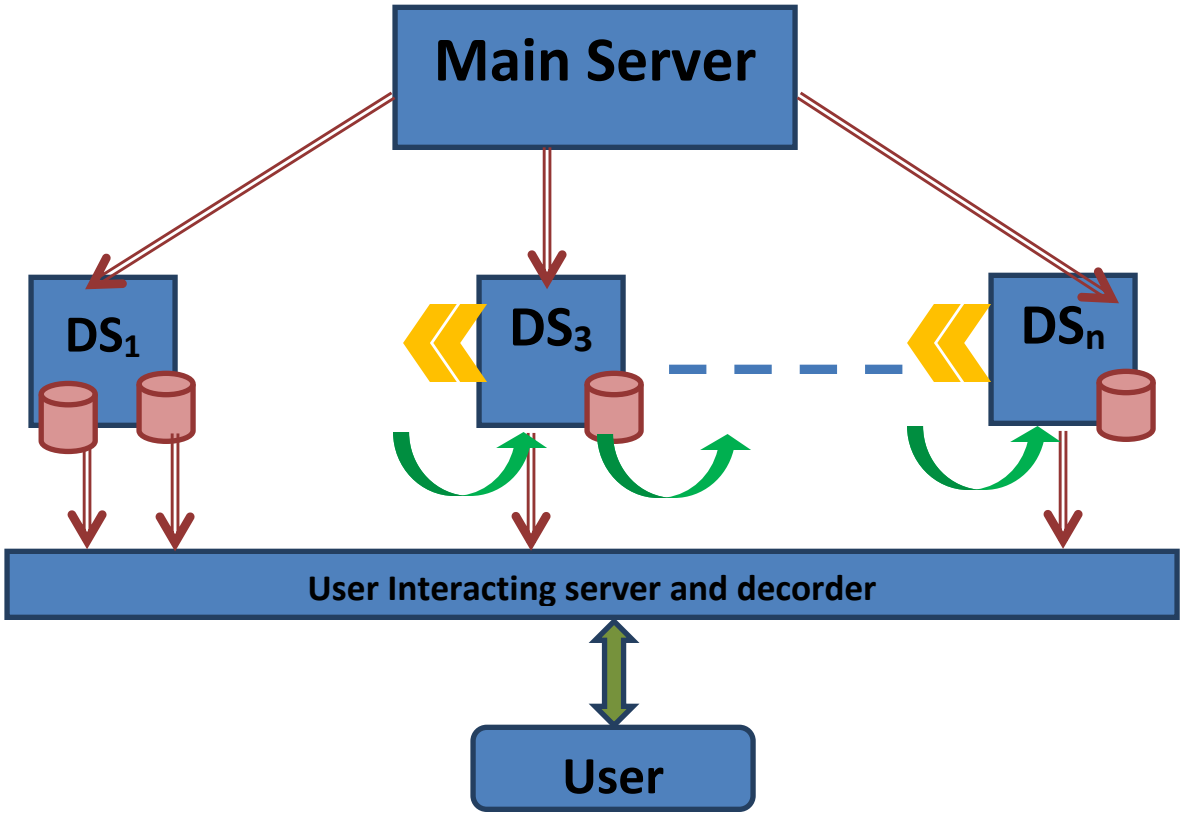

Figure 3.5: Main server created a new drop and passed on to server1

In this way any number of server failures can be tolerated in the system described above. The user will be able to fetch its data in any case.

CHAPTER 4

RESULTS

- LIMITATION OF SHAMIR SECRET
- RESULTS OF REED SOLOMON CODING
- FOUNTAIN CODE COMPLEXITY
- COMPARISON BETWEEN RS CODES AND FOUNTAIN CODES

As we have already described in chapter 2 about the three erasure coding schemes which are Reed Solomon coding, Fountain Code and Shamir secret codes. Reed Solomon coding is the most widely used scheme so far. But as per our research work we have found that fountain Codes over perform the Reed Solomon Coding in the time complexity and the implementation cost.

In this section we will present a comparison of the erasure coding schemes with some experimental results and will conclude that which scheme is better.

4.1 Limitation of Shamir Secret Algorithm

Shamir secret Algorithm[31, 32] suffers from two basic problems .The very first problem with Shamir secret algorithm is that, in case of any alteration to the data by some intruder with the motive of wrong retrieval of data, will still lead to fetching of the data through secret algorithm. But the data fetched will be wrong and the user will not be aware of this. This happens because Shamir secret algorithm works on the principle that the message will be recovered in presence of k number of secrets [32]. So, rather than getting the wrong data, user should be reported that its data has been altered. Reed Solomon coding actually takes care of this. Moreover the time complexity of Shamir Secret Algorithm is O (n^3).

The Shamir Secret Algorithm Complexity Graph is shown in Figure 4.1. The graph drawn on next page shows the time taken in response to the source length for various inputs.

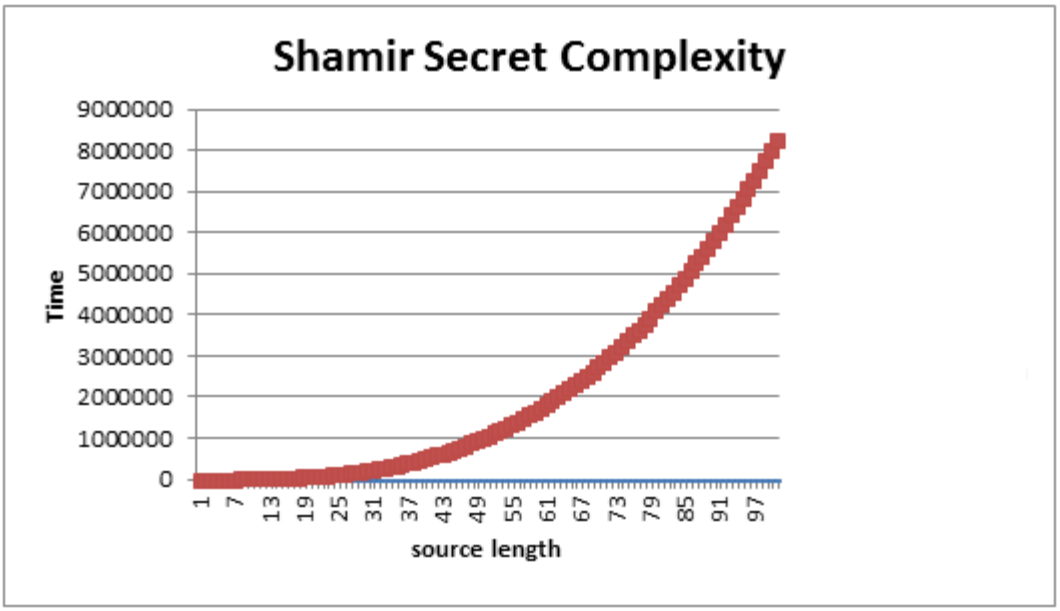

Figure 4.1: Shamir Secret Complexity

As it can be observed from the above graph that the time taken for encoding the packet increases in proportion to that of the source length on a scale of y=f(x) function. Shamir Secret algorithm can prove to be a very time taking algorithm if the source length of the message is very large. So practically this algorithm cannot be implemented on cloud architecture because of very high time complexity.

4.2 Results of Reed Solomon coding.

There are plenty of methods for implementing Reed Solomon coding. A lot of work has already been done in this field which aims at improving the encoding and decoding complexity of RS coding scheme. For our purpose we have considered the Vandermonde implementation of RS coding scheme whose encoding and decoding complexity has been discussed below.

Encoding is estimated by pre calculating the generator matrix and then multiplying the source vector(k-elements) by generator matrix(k*n elements) [26].The complexity of the pre-calculation of the generator matrix can be estimated as the complexity of the multiplication of the inverse of a Vandermonde matrix by n-k vectors (i.e., the last n-k columns of V{k,n}).Since the complexity of the inverse of a k*k Vandermonde matrix by a vector is $O(k * (\log(k))^2)$, the generator matrix can be computed in $0((n-k)* k * (\log(k))^2))$ operations. When the generator matrix is pre-computed, the encoding needs k operations per repair element (vector-matrix multiplication).

Encoding can also be performed by first calculating the product $(s * V\{k,k\})^{-1}$ and then by multiplying the result with V{k,n}. The multiplication by the inverse of a square Vandermonde matrix is known as the interpolation problem and its complexity is $O(k *(\log(k))^2)$. The multiplication by a Vandermonde matrix, known as the multipoint evaluation problem, requires O((n-k) * log(k)) by using Fast Fourier Transform, as explained in [34]. The total complexity of this encoding algorithm is then $O((k/(n-k)) * (\log(k))^2 + \log(k))$ operations per repair element.

	(30,10)	(250,50)	(250,100)	(250,125)
Vandermonde	0.007 ms	0.620 ms	2.577 ms	3.143 ms

Table 4.1

The table 4.1 shows the time taken to evaluate Vandermonde Matrix for different values of (n,k).

Further all the initialization, encoding and decoding speeds are evaluated on a Windows operating system, featuring a 2.4 GHz Intel Core i5 CPU.

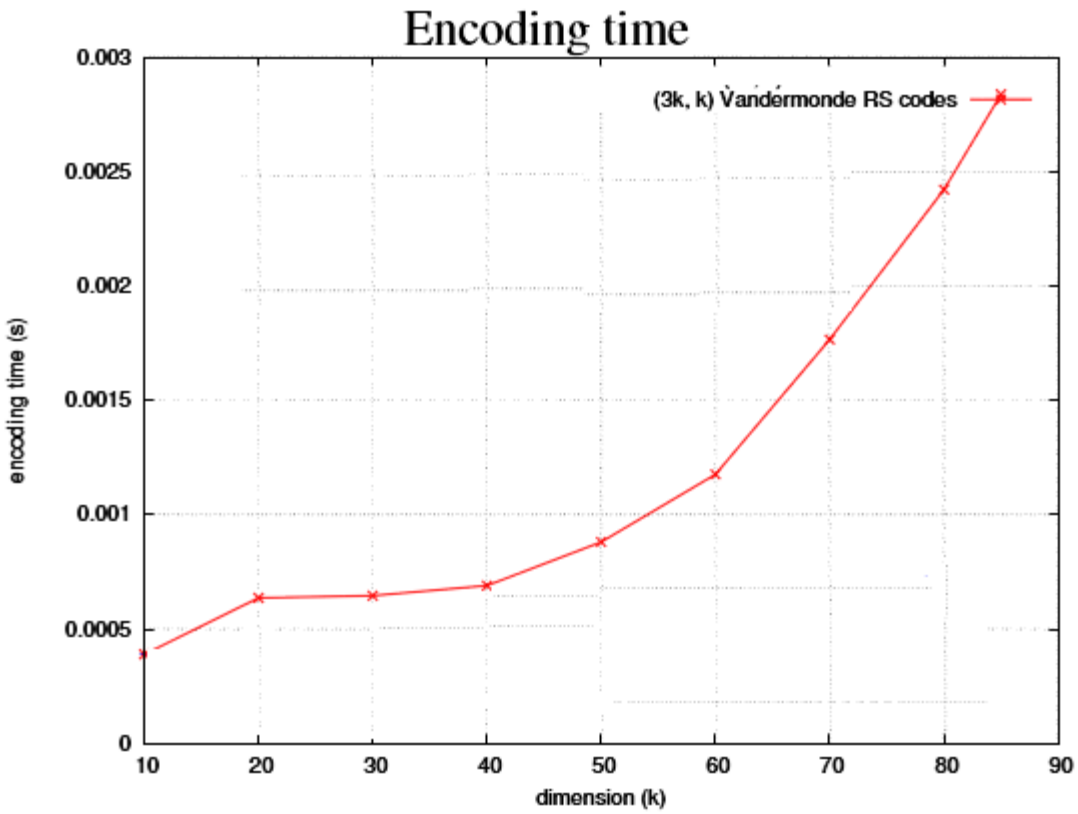

Figure 4.2: Encoding complexity of RS(3k,k) code

The Figure 4.2 showcase the encoding time of Reed Solomon coding for different dimensions of K. Note that above graph is being generated for (3k,k) configuration of Reed Solomon Coding, i.e. we can recover the message if maximum of one third of the total components are being corrupted or altered.

The above graph basically depicts that if there are total 3k distributed servers and out of that if k servers fails, then how much time is needed in that case for encoding and decoding of the complete data, for different values of k.

For example as seen from the above graph, if we have 270 servers and out of that if 90 servers fail then it will take around 0.003 seconds to encode.

The Figure 4.3 shows the decoding complexity of Reed Solomon Coding of the Reed Solomon Coding.

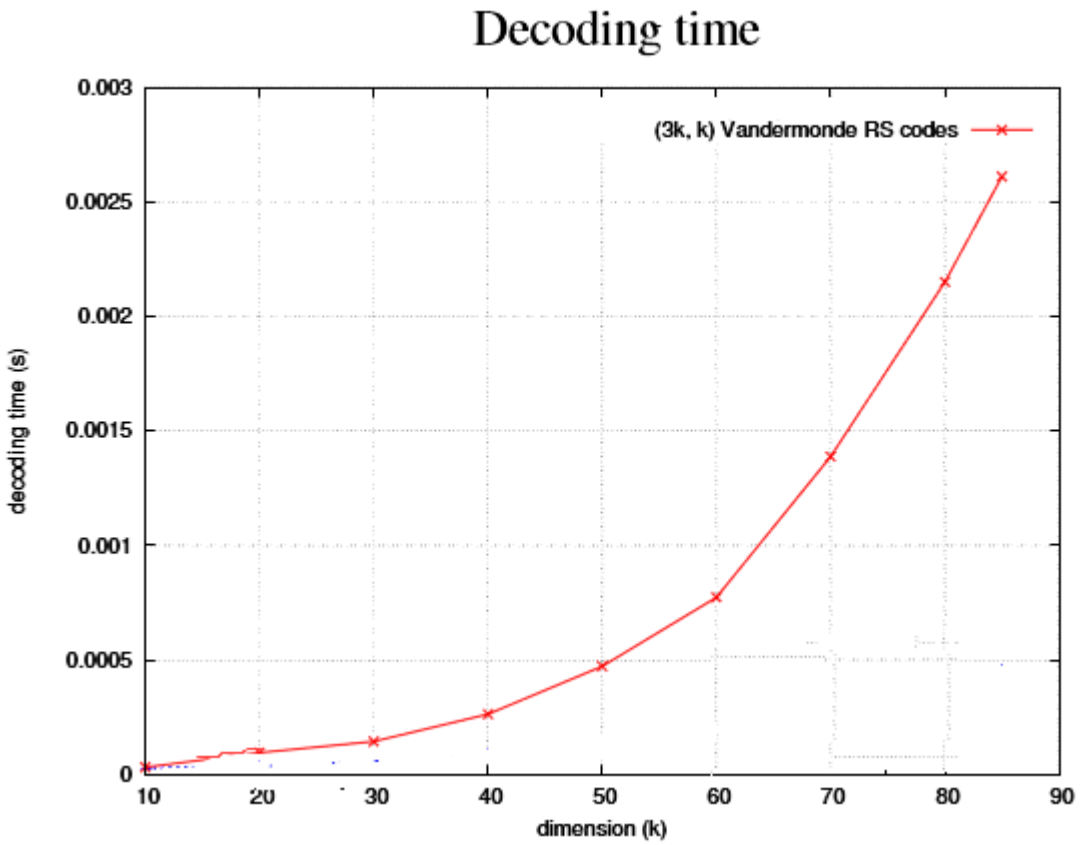

Figure 4.3 : Decoding complexity of RS(3k,k) code

-

4.3 Fountain Code Complexity

Fountain Code has linear time complexity unlike the Reed Solomon Coding. As already explained in previous section (2.3) that complexity of Fountain code depends on certain parameters. In this section we present the complexity variation with the variation of those factors and we will also present the overall encoding and decoding time complexity of fountain code.

As we have discussed in section … there are many types of fountain Codes like LI Raptor Codes, LR raptor codes, Luby Transform Code. We have shown the encoding complexity of all the techniques in Figure 4.4.

Encoding Complexity of Fountain Codes

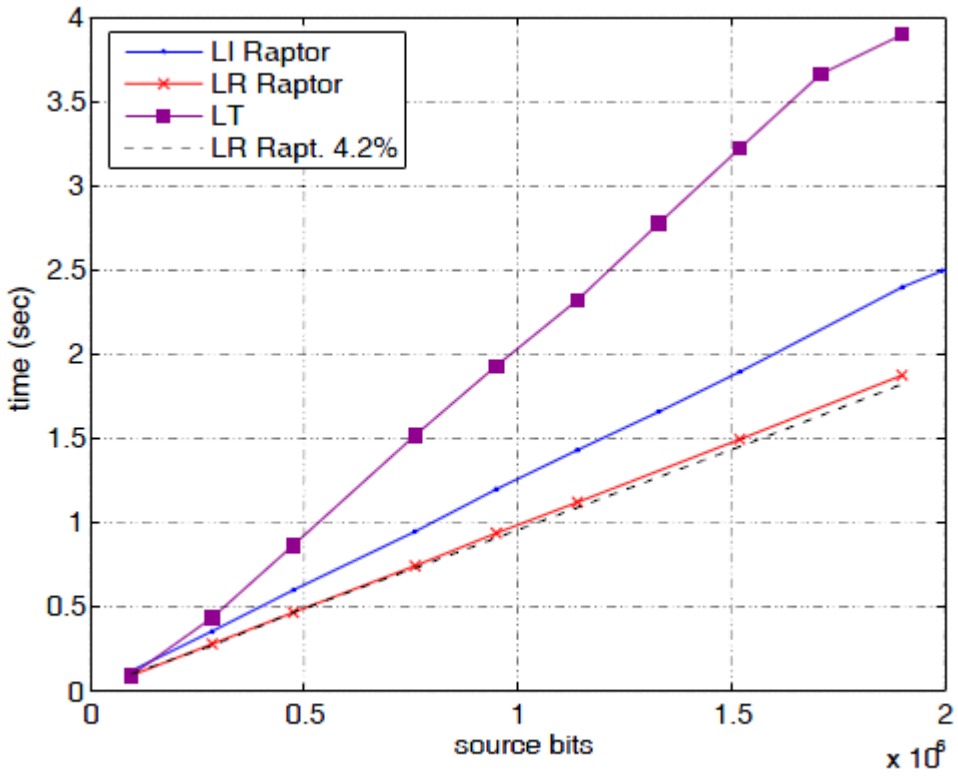

Figure 4.4: Fountain code encoding complexity

As it can be seen from Figure 4.4 that LR raptor codes perform best among all the existing fountain codes. So, we suggest using the LR raptor codes for implementing the algorithm that has been described in section 2.4

Figure 4.5 will show the Decoding Complexity of different fountain codes.

Decoding Complexity of Fountain Code

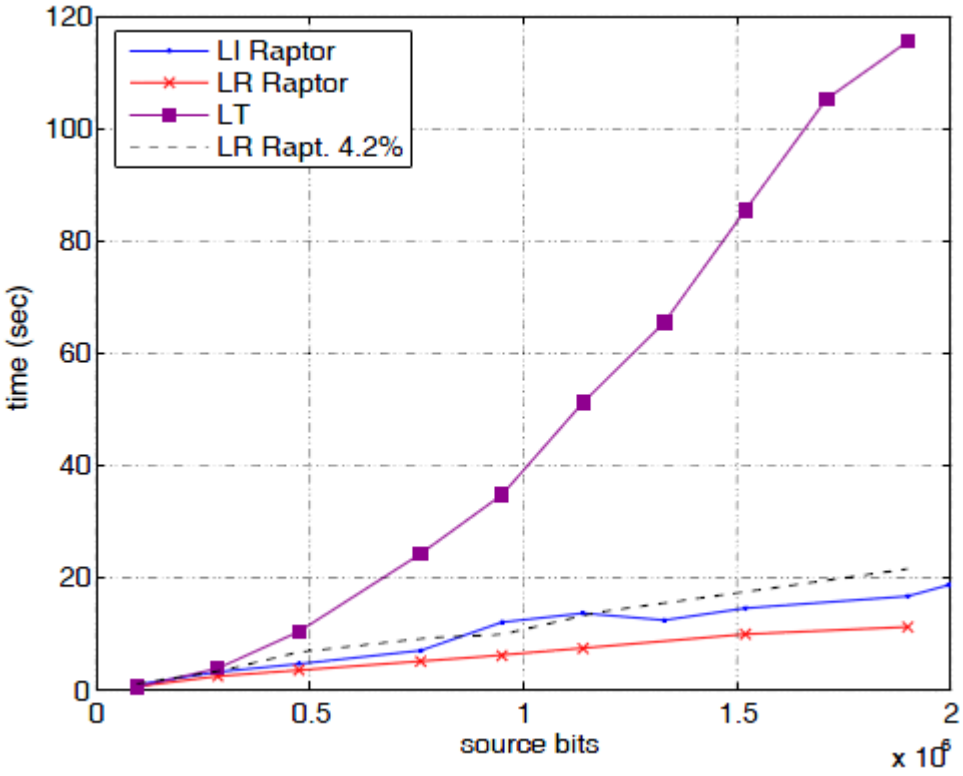

Figure 4.5: Decoding complexity of Fountain codes

4.4 Comparison between Reed Solomon and fountain Codes

This section presents a comparison between Reed Solomon Coding and fountain Codes on the basis of time taken with respect to the source length.

An overhead of some extra space utilization has also been taken care of for both the schemes.

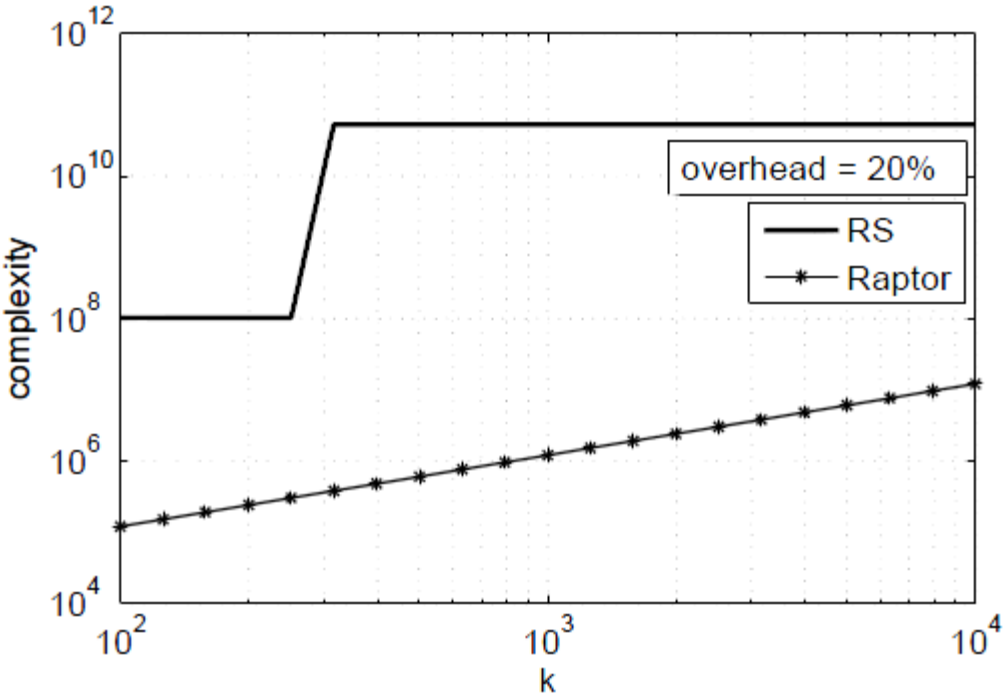

Figure 4.6 : RS and Fountain code comparison

In the above graph, assuming a 64-bit processor, one unit of complexity corresponds to one addition, one multiplication or 64 XOR operations. The curves confirm the higher decoding complexity of the RS code. The curve of the RS code increases step-by-step, as the decoding complexity is a function of the size 2q of the GF; we restrict our attention to values of q that are multiples of 8.

A RS code with q = 8 (for which the maximum value of k is 212 when considering a 20% overhead) has the same decoding complexity as a Raptor code with k = 89105 information packets. It is clear that for the same decoding complexity, the Raptor codes are allowed to be much longer than the RS codes.

The more the number of the operations, the more complex and more time the algorithm will take in order to execute.

Figure.4.6 shows a relation of source length and number of XOR operations between raptor code and reed Solomon code for various values of n and k.

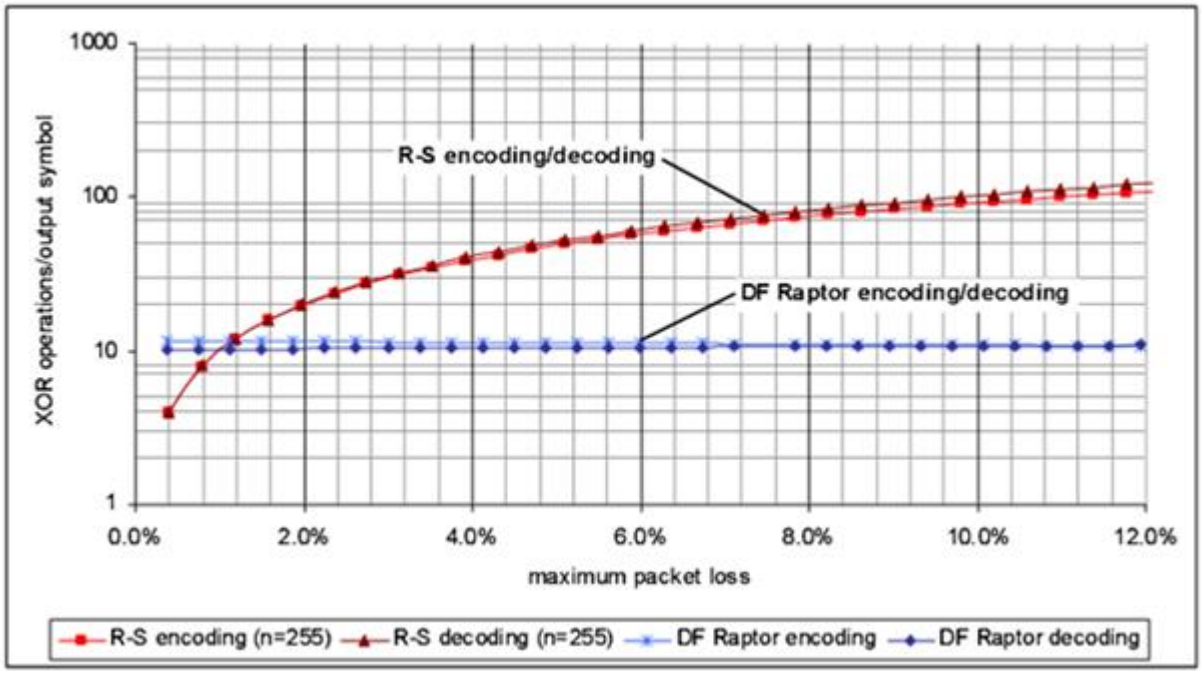

Figure 4.7: RS and Fountain code complexity for different configurations

As it can be observed from Figure 4.7 that as the percentage of packet loss increases, the number of XOR operations in Reed Solomon coding also increases. In contrast to that, in case of raptor codes, the number of XOR operations is almost constant in each case. This is probably because Raptor codes are capable of generating infinite number of drops which are used for packet retrieval.

4.5 Comparison of Vandermonde REED Solomon ,Couchy Reed Solomon and Fountain Codes

This section will present a comparison of 2 different implementation of Reed Solomon Coding (Vandermonde implementation and Cauchy implementation) and Fountain codes (Raptor Codes)

We have taken source length as the parameter and calculated the time taken to encode and decode the source.

Encoding Complexity

Source length(kb)	Time(sec)		
	Vandermonde implementation	Couchy Impl.	Fountain Code
250	9	4.6	0.06
500	39	19	0.18
1024	150	93	0.29
2048	623	442	0.57
4096	1002	1717	1.01
8192	1567	6994	1.93
16384	3211	30802	3.93

Table 4.2

As observed from Table 4.2, increasing the source length increases the time taken to encode and decode. In case of Vandermonde implementation of RS code, the time taken is very large as compare to Couchy and Fountain Codes. For a source length of 250kb, Vandermonde implementation is taking 9 sec and for the same Couchy implementation is taking just 4.6 seconds(nearly half to that of Vandermonde implementation).On the other hand, fountain Codes are almost taking negligible time as compared to both of the other techniques(even less that 1/10 of Vandermonde time).Similarly for other inputs also Fountain Codes are outperforming the RS codes. Figure 4.8 shows a graph that has been plotted for the above table. By carefully observing the graph, it can be seen that Fountain Codes are showing a linear time complexity and are taking the least time as compared to RS codes.

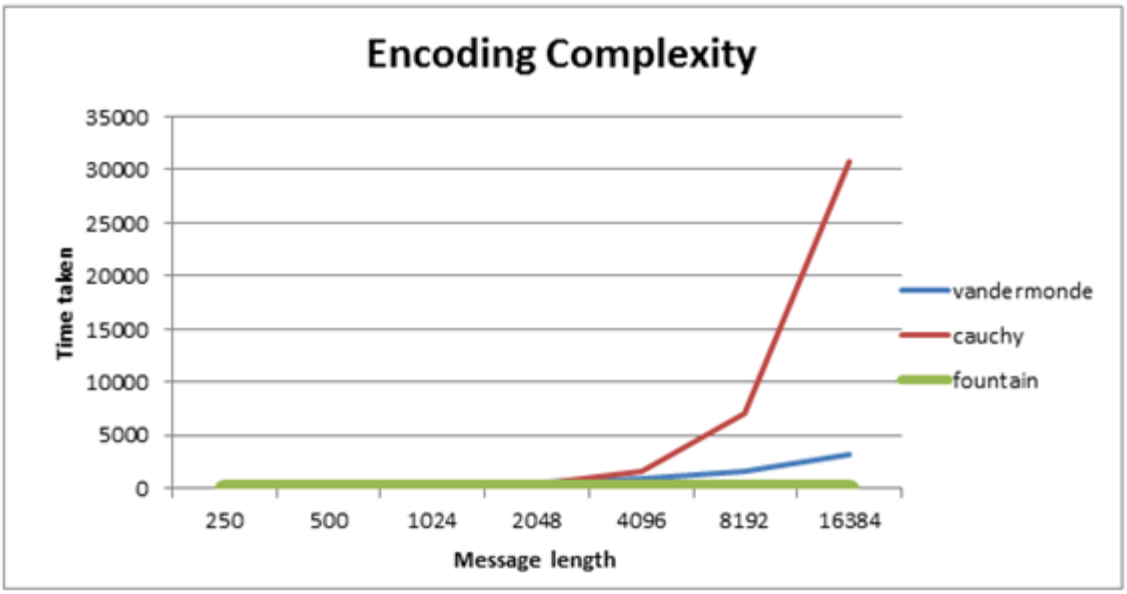

Figure 4.8: Vandermonde, Couchy and raptor encoding complexity

Similarly as like the encoding complexity, decoding time has also been recorded on applying certain input length for RS codes and Fountain Codes and the results are shown in table 4.3.

Decoding Complexity

	Time(sec)		
Source length	Vandermonde implentation	Couchy Impl.	Fountain Code
250	11	2.06	0.06
500	32	8.4	0.09
1024	161	40.5	0.14
2048	1147	199	0.19
4096	1596	800	0.4
8192	1869	3166	0.87
16384	30596	13269	1.75

Table 4.3

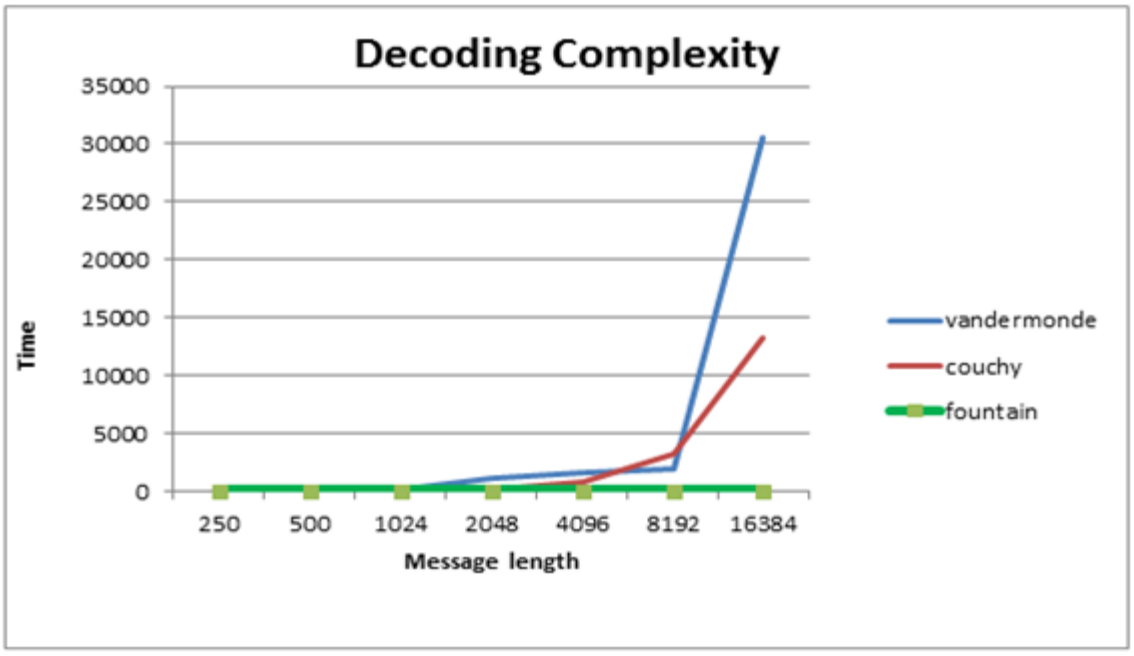

Figure 4.9: Vandermonde, Couchy and raptor decoding complexity

A careful analysis of all the results shown in this chapter justifies the use of Fountain codes over any other Erasure coding scheme for implementing in the Cloud architecture. Especially when compared with the tradition technique of RS codes (which are still being used practically), Fountain codes provides proves to be well deserved replacement over the RS codes.

CHAPTER 5

CONCLUSION

Although Cloud computing is seen as a new phenomenon which is set to revolutionize the way we Internet is used, there is much to be cautious about. There are plenty of new technologies emerging at a rapid rate, each with technological advancements and with the potential of making human's lives easier. However, we must be very careful to understand the security risks and challenges posed in utilizing these technologies. Cloud computing is no exception.

The current work has been focused on the problem of data security in cloud, which is primarily a distributed storage system. To make sure that user's data is stored correctly in cloud, we proposed an effective and flexible distributed scheme. We have used the erasure-correcting code in the file distribution preparation to provide redundancy and guarantee the data dependability and availability in case of any failure. By using the HMAC which is a kind of homomorphic token with distributed verification of erasure coded data, our scheme combines the storage correctness and data error localization, i.e., whenever data corruption has been detected during the storage verification across the distributed servers, we can almost guarantee the data retrieval and simultaneous identification of the misbehaving server(s). With some experimental results, we have also justified the use of Fountain Codes as an erasure coded scheme in distributed architecture .The scheme implementation would perform best as compared to other techniques (Reed Solomon Coding). It has also been shown that our scheme is highly efficient and resilient to Byzantine failure, malicious data modification attack, and even server colluding attacks.

5.1 Future work

1. Data storage security in Cloud Computing, an area full of challenges and of paramount importance, is still not fully explored, and many research problems are yet to be identified.

In future a complete model for data storage security can be given through which user fully trust the cloud for safeguarding its data and resources.

3. There are several possible directions for future research on this area. The most promising one could be the model in which public verifiability is enforced. Public verifiability, supported in [6] [4] [17], allows TPA to audit the cloud data storage without demanding user's time, feasibility or resources.

4. One more problem that can be worked out in future could be the dynamic data updating and deletion in the cloud.

The model presented in this thesis is the initial step and still needs more modifications; however it can provide the basis for the deeper research on security deployment of cloud computing for the research community working in the field of Cloud Computing. Still a complete solution with data updating in case of Fountain Codes can be figured out.

REFERENCES

1. Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M. "A Break in the Clouds: Towards a Cloud Definition," *ACM SIGCOMM Computer Communication Review,* 39(1), 2009, pp : 50-55.
2. Armbrust, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, A., and Zaharia, M.l" A view of cloud computing," *Communications of the ACM, 2010*, pp : 50-58
3. Buyya , "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems.* 25(6), 2009, pp : 599-616.
4. Knorr, E. "What cloud computing really means," (2010), Available at: *http://www.infoworld.com.d.cloud-computing/what-cloud-computing-really-means-031*
5. Velte, T., Velte, J., Elsenpeter, R," Cloud Computing: A Practical Approach,"*McGraw-Hill Osborne Media.* (2009).
6. Wood, T., Prashant, S., Venkataramani, A. and Yousif, M.. "Black-box and Gray-box Strategies for Virtual Machine Migration," *4th USENIX Symposium on Networked Systems Design & Implementation,* (2007), pp *:* 229-242.
7. Shih, E., Bahl, P. and Sinclair, M, "Wake on wireless: An event driven energy saving strategy for battery operated devices," *Proceedings of the 8th annual international conference on Mobile computing and networking,* (2002), pp : 160-171.
8. Vouk, MA. "Cloud Computing - Issues, Research and Implementations," *Journal of Computing and Information Technology, 16(4),* (2008), pp : 235-246.
9. "Cloud Computing – Demystifying SaaS, PaaS and IaaS by Cloudtweaks." Available at: *http://www.cloudtweaks.com/2010/05/cloud-computing-demystifying-saas-paas-and-iaas/*
10. Mather, T. Kumaswamy, S. Latif, S. "*Cloud Security and Privacy: An Enterprise Perspective on Risks and Compliance," O' Reilly media,* (2009).
11. Brodkin, J. Gartner: "Seven cloud-computing security risks," *Networkworld*, (2008), Available at:http://folk.ntnu.no/oztarman/tdt60/cloud%20computing/3%20Cloud_Computing_Security_Risk.pdf
12. Pearson, S. "Taking Account of Privacy when Designing Cloud Computing," *Software Engineering Challenges of Cloud Computing, 2009.* CLOUD '09, (2009), pp : 44-52.
13. Howe, Denis (ed.). "Redundant Arrays of Independent Disks from FOLDOC," *Free On-line Dictionary of Computing (Imperial College Department of Computing).* Retrieved 2013-03-10.

14. David A. Patterson, Garth Gibson, and Randy H. Katz," A Case for Redundant Arrays of Inexpensive Disks (RAID)," *University of California Berkeley.* 1988.
15. Wikepedia.com "RAID" available at http://en.wikipedia.org/wiki/RAID.
16. Search Storage.com "RAID alternatives: Will erasure codes rule?" available at http://searchstorage.techtarget.com/tip/RAID-alternatives-Will-erasure-codes-rule
17. MacKay, D.J.C, "Information theory, inference, and learning algorithms," *Cambridge University Press, 2003*, Available from www.inference.phy.cam.ac.uk/mackay/itila/
18. Luby, M, "LT codes," *Proc. 43rd Ann. IEEE Symp. on Foundations of Computer Science*, 16–19 November 2002, pp : 271–282
19. Shokrollahi A "Raptor codes," *Technical report, Laboratoire d'algorithmique, Ecole Polytechnique Fiederale de Lausanne, Lausanne, Switzerland,* 2003. Available from algo.epfl.ch/
20. Yaoguo Ma ,Jin Xu , Jinqiong Tang ,Keping Long ," Truncated Robust Soliton Distribution on LT Codes for Multimedia Transmission," *Multimedia Technology (ICMT), 2010 International Conference .*
21. Nick's Blog, "Damn Cool Algorithms: Fountain Codes," available at http://java.dzone.com/articles/algorithm-week-fountain-codes .
22. G. A. Gibson, L. Hellerstein, R. M. Karp, R. H. Katz, and D. A. Patterson," Failure correction techniques for largedisk arrays," *Third International Conference on Architectural Support for Programming Languages and Operating Systems*, pp : 123–132, Boston, MA, April 1989.
23. M. Blaum, J. Brady, J. Bruck, and J. Menon.,EVENODD, "An optimal scheme for tolerating double disk failures in RAID architectures," *21st Annual International Symposium on Computer Architecture*, pp. 245—254, Chicago,IL, April 1994.
24. W. W. Peterson and E. J. Weldon, Jr ,"Error-Correcting Codes, Second Edition," *The MIT Press, Cambridge*, Massachusetts, 1972.
25. D. Wiggert, "Codes for Error Control and Synchronization," *Artech House, Inc., Norwood, Massachusetts,* 1988.
26. James S. Plank, "A Tutorial on Reed-Solomon Coding for Fault-T olerance in RAID-like Systems," *Technical Report CS-96-332 Department of Computer Science University of T ennessee.*
27. CE 341/441, "LECTURE 6 LAGRANGE INTERPOLATION" available at www3.nd.edu/~jjwteach/441/PdfNotes/lecture6.pdf
28. wordpress.com ,"Numerical Method," available at http://jayemmcee.wordpress.com/lagrange-polynomial-interpolation/

29. springer.com "Chapter2 Finite Field Arithmetic," available at https://www.google.co.in/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&cad=rja&ved=0CEYQFjAC&url=http%3A%2F%2Fwww.springer.com%2F%3FSGWID%3D5-102-45-110359-0&ei=7butUc6eJOSdiQKA7oCwCg&usg=AFQjCNEBQezlDhtPomQ8BIUKth2HtpsuLg&sig2=3yK97_cyEAnmTUzExSiVsA&bvm=bv.47380653,d.cGE
30. Trinity College, "Course 373-Finite fields," available at pet.ece.iisc.ernet.in/sathish/FiniteFields.pdf.
31. Nanyang Technical University, "lecture Notes on Secret Sharing," available at www3.ntu.edu.sg/home/carlespl/arc02v03.pdf.
32. Liu, C. L. , Introduction to Combinatorial Mathematics, New York: McGraw-Hill 1968.
33. Dawson, E.; Donovan, D. , "The breadth of Shamir's secret-sharing scheme," *Computers & Security* **13**, 1994, pp : 69–78,
34. Gao, Shuhong. "A new algorithm for decoding reed-solomon codes," *Communications, Information and Network Security*. Springer US, 2003. pp : 55-68.
35. Knuth, D. E. , *The Art of Computer Programming*, II: Seminumerical Algorithms (3rd ed.), Addison-Wesley,1997, pp : 505.