Czech Technical Univeristy in Prague
Master of Science Degree in Electronics, Communication and Multimedia Engineering
Faculty of Electrical Engineering

MASTER THESIS

Privacy for Secure Distributed Storage Networks

Author:	Enio Marku
Matriculation Number:	437928
Academic Tutor:	Ph.D. Tomas Vanek
Tutor:	Ph.D. Thomas Lorünser
Deadline:	9^{th} of January, 2017
Date of Birth:	02^{nd} of December, 1992

ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Marku** Jméno: **Enio** Osobní číslo: **437928**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra teorie obvodů**

Studijní program: **Komunikace, multimédia a elektronika**

Studijní obor: **Komunikační systémy**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Zajištění soukromí v bezpečných distribuovaných síťových úložištích.

Název diplomové práce anglicky:

Privacy for Secure Distributed Storage Networks

Pokyny pro vypracování:

Study methods of the privacy assurance in the cryptographic storage networks based on secret sharing. Focus on sensitive data retrieval and the possibility of hidden access controls in the context of user interaction in the open source Archistar framework. Do a survey on existing secret sharing schemes as well as their adoption and possible performance improvements for the Archistar project. In hands-on part implement designed protocols and recommended enhancements in suitable software and do measurement of its performance in real world configurations.

Seznam doporučené literatury:

[1] T. Loruenser, A. Happe, D. Slamanig (2014). “ARCHISTAR – A framework for secure distributed storage”. GNU General Public License. http://ARCHISTAR.at
[2] I. Goldberg, Improving the Robustness of Private Information Retrieval, Proc. of 2007 IEEE Symposium on Security and Privacy (Oakland 2007), May 2007, available at: http://www.cypherpunks.ca/~iang/pubs/robustpir.pdf [online]
[3] C. Devet, I. Goldberg, N. Heninger, Optimally Robust Private Information Retrieval, 21st USENIX Security Symposium, August 2012, available at: http://www.cypherpunks.ca/~iang/pubs/orpir-usenix.pdf [online]
[4] W. Lueks, I. Goldberg, Sublinear Scaling for Multi-Client Private Information Retrieval, 19th International Conference on Financial Cryptography and Data Security, January 2015, available at: http://www.cypherpunks.ca/~iang/pubs/slspir-fc15.pdf [online]

Jméno a pracoviště vedoucí(ho) diplomové práce:

Ing. Tomáš Vaněk Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **21.12.2015** Termín odevzdání diplomové práce: **09.01.2017**

Platnost zadání diplomové práce: **30.09.2017**

Podpis vedoucí(ho) práce	Podpis vedoucí(ho) ústavu/katedry	Podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

. | Datum převzetí zadání

Podpis studenta

Statement

On the basis of this statement I declare to have issued, produced and compiled this master thesis solely by myself. All means and sources used are quoted accordingly.

Declaration Date:	5^{th} of January, 2017
Location:	Prague

Signature:

Abstract

In a cloud scenario outsourcing data to third party providers enables processing benefits and significant cost. As a result there is a high tendency of different companies to outsource their data and services to third party service providers. On the other hand, there exists a high concern when it comes to privacy of the outsourced data. Since in a storage system a side channel attack is possible, it can cause the leakage of sensitive information, outsourcing comes with an expensive price, thus we need to develop algorithms which can prevent such information to be known by cloud providers. The security issues here are: malicious insiders, data locality, lack of control of the outsourced data, legislation issues, data and service availability, vendor lock-in etc.

The main goal of the thesis is to do a research over the existing algorithms which are used by cloud computing platforms in order to prevent such leakage information and to develop and implement a suitable protocol which can hide such sensitive information from cloud providers. Private Information Retrieval Protocol is one way to achieve that. Private Information Retrieval is a database technique that uses idea from cryptography to protect the privacy of users by hiding the details of their queries from the operators of the database. There are several PIR Protocols but in this thesis the focus is on PIR Protocol which uses Shamir Secret Sharing Threshold Scheme.

Index Terms:

PIR (Private Information Retrieval), SSS (Shamir Secret Sharing), AWS (Amazon Web Service)

Abstract

V outsourcingu dat mrak scénář poskytovatelů třetích stran umožňuje výhody zpracování a značné náklady. V důsledku toho je vysoká tendence různých společností zadávat své údaje a služby třetím poskytovatelům služeb. Na druhé straně existuje velké obavy, pokud jde o soukromí externě dat. Vzhledem k tomu, v úložném systému útok postranním kanálem je to možné, může dojít k úniku citlivých informací, outsourcing přichází s drahou cenu, a tak musíme vyvinout algoritmy, které mohou zabránit takové informace, které mají být známy poskytovateli cloudu. Tyto bezpečnostní problémy jsou zde: škodlivé zasvěcenci, data lokalita, nedostatečné kontrole outsourcovaných údajů, legislativa otázky, dat a dostupnost služeb, dodavatele lock-in atd. Par Hlavním cílem této práce je dělat výzkum přes existujících algoritmů, které používají cloud computing platforem, aby se zabránilo takovému úniku informací a vyvinout a zavést vhodný protokol, který může skrývat takové citlivé informace od poskytovatelů cloudových. Soukromé získávání informací Protocol je jedním ze způsobů, jak dosáhnout, aby. Soukromé získávání informací je databáze, technika, která využívá myšlenku z kryptografie k ochraně soukromí uživatelů tím, že skryje podrobnosti o svých dotazů od provozovatelů databáze. Existuje několik PIR Protokoly, ale v této práci je pozornost zaměřena na PIR protokol, který nese shamirovo sdílení tajemství Threshold Scheme.

Contents

Acknowledgements

This work is carried out as fulfillment of the last requirements for the Master of Science degree in Electronics, Communication and Multimedia in the Czech Technical University in Prague (CVUT). The preparation of the thesis is performed at Austrian Institute of Technology (AIT). Firstly, I would like to thank both my supervisors Ph.D. Tomas Vanek and Ph.D. Thomas Lorounser for giving me the possibility to work with them on this project. I am grateful to them for their time spent on me, technical support, advices and new ideas during the project. It has been a great help for my graduation process and hopefully for my future career as well. The experience gained during my work on thesis is valuable to me not only from technical aspect, but also for my personal and cultural aspects. A salutation deserves also the research group of Archistar project for providing a friendly environment making things nicer and easier for me.

I dedicate this work to my wonderful family, to which being thankful is not enough, but still. Thank you for comprehensive and continuous support during my whole life, especially during my time spent abroad.

Enio Marku
January 2017

Chapter 1

Introduction

"Knowledge is power. Information is liberating. Education is the premise of progress, in every society, in every family."

Kofi Annan

1.1 Motivation

Since the idea of the Cloud Computing comes into practice everything related to the usage of internet has become a relief. To illustrate this some examples are shown. Cloud Computing means computing based on the internet. It allows you to get your application online, to store your files online. Before the idea of the cloud, storing your files was much more complex. To do so, firstly, you had to save them into a personal computer and to take them with you, you would have to save them into a CD or flash drive. Nowadays, with cloud being part of our life we can save files only and can access them from anywhere.

Some of the best cloud storage services are Gmail, Google docs., Drobbox, pCloud. Some of the Cloud Platforms are Windows Azure, Amazon Web Services. Later the security algorithms and protocols these platforms are using to keep the information of their clients safe, will be discussed. Most of the companies nowadays are moving into cloud and use at least one of the cloud services. Some of the key points of such an involvement into cloud computing are:

- flexibility
- cost
- complexity of owning and operating computers and networks
- security

One important topic of the thesis is privacy. When it comes to the cloud, privacy is one of the most important topics. As the industry of cloud computing is currently experiencing tremendous growth and each day more business are using cloud, the need of preventing information leakage is becoming a big concern for cloud providers.

1.2 Problem Statement

When we upload our data in a cloud service such as Drobbox our data is encrypted and then outsourced by a third party. Even if the data is encrypted we are not entirely safe that our data can not be stolen for some reasons listed below:

1. Different cloud providers use different encryption algorithms but none of them is theoretically unbreakable.
2. A cloud storage provider can learn which data is accessed, how often and what action a client can take.
3. A cloud storage providers can learn who has access to this data or who accesses the data .
4. Also a cloud storage provider can learn from where the client is accessing the data and how much data is stored.

The last three points are information that can be revealed by using a side channel attack, and such side channel exists in a storage system. Such information are called meta-data information. So in some scenarios revealing any of these meta-data information may already be problematic, even if the content data is unknown.

1.3 Aim of the thesis

The overall aim of the thesis is to find and implement a suitable protocol which can hide these meta-data information. Then, this portocol is deployed in an existing cloud platform such as AWS in order to do some performance measurements and see how it works and hide meta-data information.

1.4 Methodological Approach

This thesis consists of two parts: The first part is a theoretical part and the second part is a practical one.

The first part of the thesis consists of Chapter 2 and 3. In chapter 2 is given an overview of Cloud and what are the benefits of using Cloud. Later in this chapter, data beyond privacy techniques used in cloud platforms such as Amazon Web Service, are discussed. In chapter 3 a mathematical description of Shamir

Secret Sharing threshold scheme is given and later in this chapter are described the current PIR Protocols and their advantages and disadvantages. Since our work regards on PIR Protocol based on Shamir Secret Sharing an overview of Galois Field is given, too. Galois Field used in this project is GF(256) and all mathematical operations are done based on that Finite Field.

The second part of the thesis consists of Chapter 4, 5. In chapter 4 a detailed description of AWS's architecture is given. Later, in this chapter the way how the application is deployed in AWS is showed. In chapter 5, the implementation details and performance measurements report of the application are given.

1.5 Related Work

In this section several research topics related to this thesis are introduced. We have divided research papers into three categories:

1. Current Privacy Enhancing Technologies.
2. Current PIR Protocols and their advantages and disadvantages.
3. Enhancement algorithms for PIR Protocol.

Note: PIR Protocol is part of the Privacy Enhancing Technology (PET).

The aim of PET is to allow users to take actions such as increasing control over their personal data, using pseudonyms, or anonymous data credentials, minimize the personal data collected and used by service providers and merchants etc, related to their personal data sent to, and used by, cloud providers.

The papers [25], [26], [27] give us an overview of privacy design strategies which encompass design patterns, data, process-oriented design strategies and privacy-enhancing technologies. In these papers a structured overview of important privacy technologies is given. Some of the privacy techniques discussed here are : Authentication, Secured Private Communication, Privacy in Database, Communication anonymity and pseudonymity, Private Information Retrieval. The paper [28] describes an hybrid PIR Protocol which combines two Pir Protocols one by Goldberg and another one by Aguilar Merchol. It is shown that this hybrid protocol is particularly beneficial when the number of records in a database is large compared to the size of the records and uses less communication than either of these component protocols. In this paper [29] a simple PIR scheme of Chor et al and Goldberg is described. The protocol implemented here is almost the same as Goldberg with the only change to the client side of Goldberg's 2007 Byzantine-robust in order to improve its Byzantine robustness from to, which is the theoretically maximum possible value. This protocol according to the taken measurements seems to be faster than the original Goldberg protocol. This was achieved by using decoding algorithms that are able to take advantage of decoding information in multiple blocks of data simultaneously, especially in scenarios when a client is interested in more than one block at a time.

In this paper [30] the robustness of private information retrieval is improved in several ways. In section 5 an enhancement of PIR Protocol based on Shamir Secret Sharing is shown and the result is that after independence to the protocol is added but neither the communication cost ,nor the number of servers, is increased.

Chapter 2

Privacy and Security in Cloud Computing

2.1 Introduction

Not many years ago people used to carry their files on floppy disks. Then later they switched to flash drives. Nowadays, we are in a new era, in the era of cloud computing. Cloud computing is the distribution of on-demand computing resources including everything from data centers to applications over the internet. Cloud computing enables us to have access on the information stored in remote servers and then manipulate it by using any device which has access over the internet, such as computers, personal computers and smartphones.

A lot of people have doubts about the advantage of saving our data in cloud. Firstly, you can access your files, projects from everywhere, from your computer at work or from your personal computer at home. Secondly, if you are an employee of IT department, cloud makes computing easier. On the other hand, the usage of dedicated servers brought a lot of problems like, cost, crashing, installation, maintenance, and full with viruses. Furthermore, what cloud computing does, is to take care of all these problems. To sum up, cloud computing reduces the cost, and the amount of work for IT employees. Cloud takes the service, technology and applications that are similar to those on the internet, and turns them into a self-service benefit.

So, due to the facilites cloud is bringing to us, nowadays, cloud is becoming a solution for small and big companies and also for individual customers. The revolution that cloud computing brought is that with internet we can access any resource that we need to.

As mentioned above the reduction of cost is as a result of sharing of storage resources and computing among the cloud's clients. On the other hand, this brings up issues of traditional security, privacy mechanisms and trust. Since the thesis is more concerned about privacy rather than security, in this chapter will be discussed the privacy mechanisms which are used in Amazon Web Service

(AWS).

Privacy here, will be referred to as the individuals have the right to 'know what is known about them', control how that information is given, be conscious of stored information about them and prevent its abuse. In other words, privacy is more than just confidentiality of information. Every individual has the right to control his or her own data, whether public, private or professional. In the online world, privacy concerns are increasingly important, especially in a cloud scenario.

The main reason why the privacy is a big concern in a cloud scenario is because the end-users consume cloud services with no information about the involved processes. This comes as a result of unknowing where the data centers are physically located or how the personal data, which will be processed, is configured. Since the cloud is a distributed system and anyone with an account can access the data stored in the cloud, data which is stored in data centers is more difficult to control and easier to be manipulated. Storing sensitive data on a data center somewhere in cyberspace could be a major threat to individual privacy. Thus a lot of questions related to the cloud security and privacy are raised. How relieable are cloud data centers? Can we trust our sensitive information to cloud providers? What happens if data gets lost?

In this chapter a detailed description of what are the ways to protect the privacy of individuals is showed. For that, in the upcoming sections, there are given detailed examples of some of the most usable PET algorithms. Also, it is explained how Amazon Web Service deployed those algorithms in order to protect the privacy of their clients and to make their environment trustworthy.

2.2 Cloud Computing

It is very important to have a clear idea of what cloud is and how it works before moving on and discussing the privacy algorithms which are deployed in AWS and Microsoft Azure cloud platforms. Moreover, the practical work consists of deploying the application in AWS, so a deep understanding of the cloud enviroment itself is required.

There is no single common definition of what cloud is. According to National Institute of Standards and Technology(NIST), the cloud computing is defined as follows [4] :

"Cloud computing is a model which enables suitable, on-demand network access to a shared pool of resources configurable computing resources (e.g.,services, networks, data centers, applications and storage) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

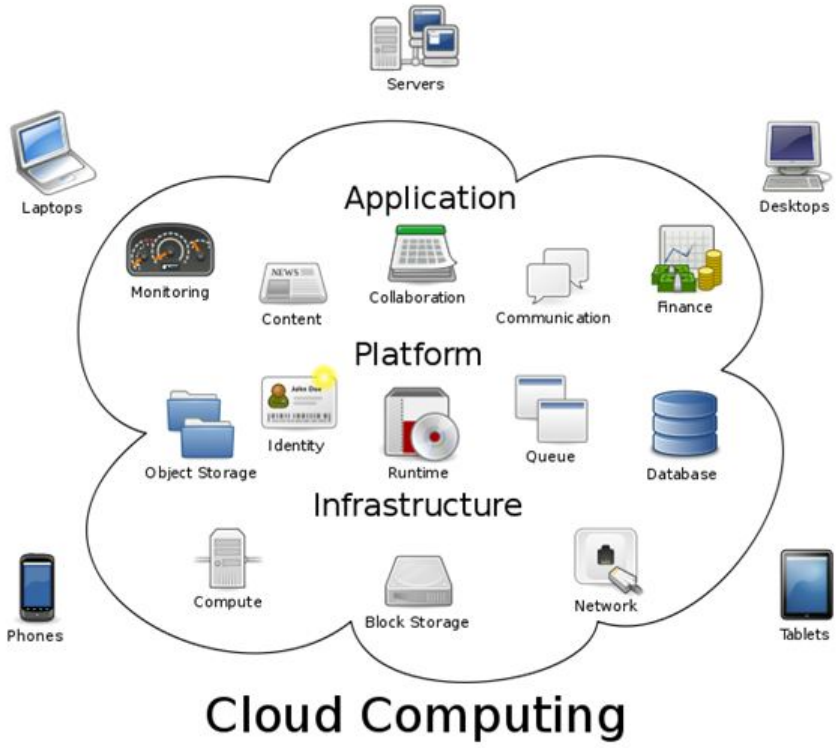


Figure 2.1: Cloud Computing architecture. Image retrieved from [10]

2.2.1 Main Characteristics

Cloud computing is defined by the following characteristics:

1. Broad network access
2. Resource pooling
3. On-demand self-service
4. Rapid elasticity
5. Measured service

Broad network access means that in order to promote use and facilitate access through heterogeneous thick or thin client platforms (e.g., laptops, tablets,

workstations and mobile phones), capabilites over the network and accessed through standard mechanisms are available. [4]

Resource pooling is about pooling the provider's computing resources in order to serve to multiple clients by using a multi-tenant model with different virtual and physical resources which according to the client request are dynamically assigned and reassigned [4]. Generally the clients have no knowledge of where exactly the provided resources are physically located, but the client may specify preferences such as countries or states for data centers in different locations.

On-demand self-service means that physical servers and network storages can be configured and ordered automatically and do not require human interaction with each service provider. [4] In contrast, in the old provisioning model the physical servers had to be configured before being used.

Rapid elasticity is a cloud computing term and it refers to the ability to provide scalable services. Clients are allowed to automatically request additional space in the cloud or other types of services. Even though the providers usually need to allocate and de-allocate resources, this is irrelevant to the client side. This is a crucial aspect of cloud technology because it means that cloud computing resources are unlimited and automatically available. This is in contrast to the older systems, where the limited storage and memory were visible to the user. [4]

Measured service is a cloud computing term and it is used as a service reference by the cloud providers in order to measure or monitor the provision of services for different reasons, including effective use of resources, billing, or overall predictive planning. According to NIST, a measured service is a setup where cloud systems may control a user or tenant's use of resources by leveraging a metering capability somewhere in the system. [4]

2.2.2 The service models

The following three service models in cloud computing are generally well-known. All of these service models allow users to run applications and store data online. Each of them offers a different level of user control and flexibility.

1. Software as a Service (SaaS)
2. Platform as a Service (PaaS)
3. Infrastructure as a Service (IaaS)

Additionally, in recent years another service model, which is called *Anything as a Service(XaaS)*, has been considered as a fourth service model. [6]

Software as a Service(SaaS). This service model allows the client to use the cloud provider's applications running on a cloud infrastructure. The client

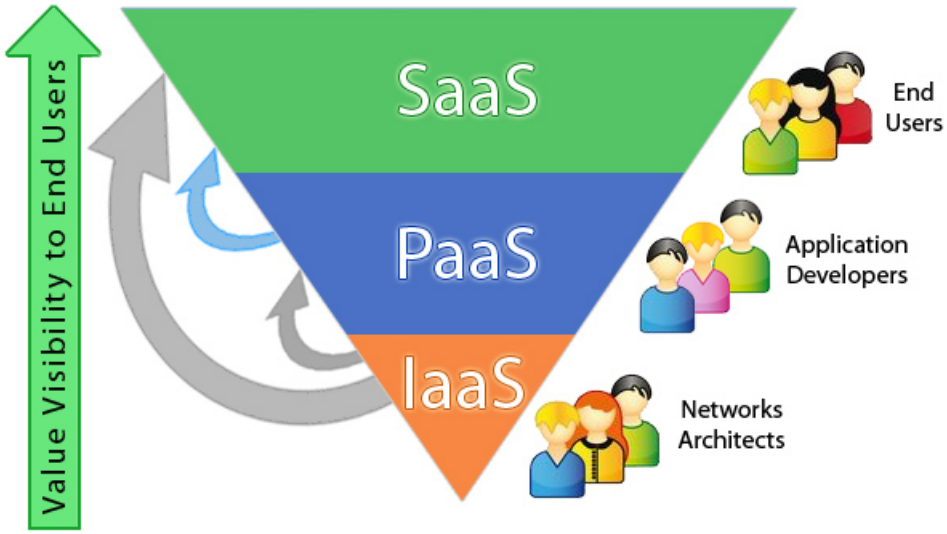


Figure 2.2: Cloud service models. Image retrieved from [7]

can access the applications provided by cloud administrator by using a web browser or a program interface. The client can not control or manage the cloud infrastructure including servers, storage, network, operating systems, or even individual application capabilities. It is allowed for a client only to control or monitor the limited user- specific application configuration settings. [4]

The benefits of using SaaS are :

1. Accessible from any computer
2. Free or paid via subscription
3. Facilitates collaborative working

Platform as a Service (PaaS). PaaS expands the benefits that SaaS in the cloud brought to the end-user. In basic terms, PaaS allows users to create their own cloud applications by providing environments and tools to them.

The benefits of using PaaS are the following:

1. Public or private deployment
2. Rapid development at low cost
3. Facilitates collaborative working

Infrastructure as a Service (IaaS). IaaS expands the idea introduced in PaaS. In basic terms, IaaS is a third-party provider that hosts software, hardware, storage and other infrastructure components on behalf of its users

and also handles tasks, including system maintenance and backup, for its users. We just need to rent the services we need to use by third-party providers. For instance if we need a database server we just rent the service and do not need to worry about the hardware, security and infrastructure.

The benefits of using IaaS are the following:

1. Cost of bandwith has gone down tremendously
2. Scalable and flexible
3. Increased infrastructure reliability

Anything as a Service (XaaS). The main idea is to expand the basic service models by making everything and anything virtually and available as a service in the cloud. Aside from the three well-known service models listed above, a number of additional models have been established [6]. Such as, Collaboration-as-a-Service, Business Process-as-a-Service,Database-as-a-Service, Strategy-as-a-Service, Network-as-a-Service and Communicationas- a-Service. The last two entries, Network-as-a-Service and Communication-as-a-Service, have been recognized by the International Telecommunication Union (ITU) as part of the cloud computing service models.

2.2.3 Deployment models

The following deployment models are defined by where the infrastructure of the deployment exists. According to NIST [4] there are four types of deployment models in cloud computing

1. Public cloud
2. Private cloud
3. Hybrid cloud
4. Community cloud

Public cloud. In this kind of deployment model there are cloud servers that are entirely hosted by external providers such as Amazon, Digital Ocean, Windows Azure. These services exist to allow clients to scale up their applications, to spin up resources on demand. In this way, the client is in charge of only administrating the server and application. The physical hardware, resource allocation and networking are all handled by the provider [11]. For instance, if the network goes down or if the server gets crashed the third-party provider is in charge of handling these problems. The benefits of using public cloud are :

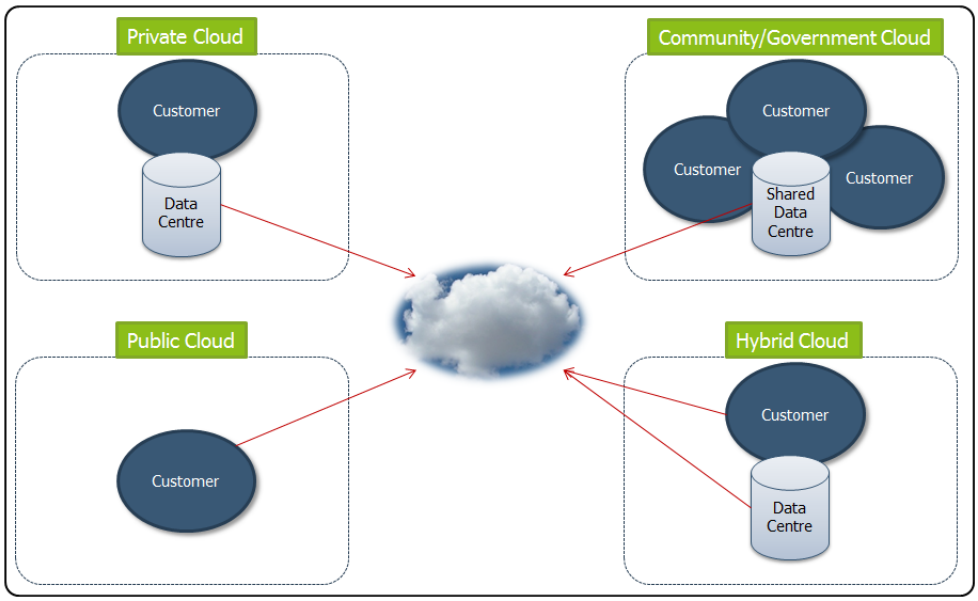


Figure 2.3: Cloud service models. Image retrieved from [13]

1. Easily accessible
2. Offer increased availabilty and scale
3. Cost savings are often passed on to clients

The availability and scalability are the main advantages of using public cloud. For instance, in AWS there are teams which are focused primarly on physical servers so that the clients do not have to worry about that. They can just focus on building their applications in deploying into the cloud.

On the other hand, there are also some downsides of using public cloud as listed below:

1. Some services, which are not supported by the provider, need to be built by the clients
2. They are dependent on the uptime and quality of the third-party provider
3. There are only a few guarantees on physical security

Private cloud. They are managed and maintained by the company itself. For example, if different organizations want to set up their own private cloud, they would essentially create their own servers and have their own physical hardware servers. They make resources available only internally, so that their applications can be deployed to their own physical controlled servers. They do

not have to go to AWS or Windows Azure, they can actually setup their own infrastructure. [11] Because of the benefits listed below, the private cloud is gaining high popularity.

1. It can ensure physical security
2. It has more monitoring
3. It has more control over hardware

The main benefit of private cloud is the *control over security* . If you have your own private cloud, you can control it when the servers are getting updated. Based on the application, this control is tremendously important.

Besides the benefits, there are also some other things we need to consider if we want to build our own private cloud. *Training and expertise are needed.* For instance, we need server administrators, networking specialists. That means that your *costs will be higher* and also *responsibility will be increased.* Now, if a server goes down, you will not rely anymore on a support contract, but you need to take your own responsibility and fix the problem.

Hybrid cloud. It comes as a combination of public and private cloud. This type of deployment model is going to be the dominant model in the near future. The main reason is that it allows the flexibility and control when it is needed, but on the other hand, it also takes the benefits of public cloud, such as cost savings, scalability and availability.

Essentially, in this case we have a public cloud and a private cloud which will communicate through some agreed protocols whether it is through sockets or Rest API. In this scenario, we will have our private cloud data which may hold our secured data and then, for the client part, like for advertising or order processing, those will be out to the public cloud.

Community cloud. This type of model is the rarest and the least known out of the other models. Essentially, it is a private cloud with several organizations. The cloud infrastructure is owned and shared between these organizations. The organizations could be companies working together, governmental organizations or research groups. Let's suppose there are three companies which will work together in a project. In that case, they will set up their own cloud and will share resources among them.

2.2.4 Advantages and Disadvantages of using cloud

The reason why companies and organizations are moving toward cloud is because cloud computing offers us some advantages as followings:

Mobility Cloud gives us the opportunity to work anytime, anywhere. It is not neccessary to go and work from your office.

Scalability Cloud gives us the opportunity to quickly build, deploy and manage our applications. Moreover, it enables us to build and expand within minutes.

Software updates All updates and maintenace will be carried out by the third-party providers.

Flexibility Depending on workload, the resources can be increased or decreased.

Collaboration Employees can get synchronized between them and work wherever they are.

Cost is reduced Now you can pay based on consumption. Payments can be paid monthly or annually.

Environmentally friendly Less hardware is required. Moreover it reduces energy consumption and carbon footprint.

Besides the advantages cloud offers to us there are also some downsides, we need to worry about when we use cloud. [15]

Security and Privacy In a cloud environment we usually save a lot of data, sometimes also sensitive data. In any discussion about data, privacy and security must be the most important concern. When we sign up in order to use the resources the third-party provider offers to us, we do not have any information where our data will be saved and who will control them. This makes the cloud environment unreliable. In order to convince the clients, the third-party providers are currently deploying the most sophisticated protocol to protect the client's data.

Downtime No cloud providers can assure us immunity to service outages.Cloud computing systems are internet based, which means that our access is dependent on our Internet connection. Like every type of hardware, cloud platforms themselves can fail for any reason.

Vulnerable to attack In a cloud platform every component of cloud is accessible by the Internet. Everything which is connected to the Internet is not secured. For that reason, even the best teams suffer severe attacks and security breeches.

2.3 Privacy techniques in Amazon Web Service

In this section, the privacy techniques will be discussed. Privacy is a very important concept when it comes to the Internet. Privacy is a fundamental human right. There is a massive power imbalance between data processing and the individuals whose data is at stake. So, in a digital world the protection of private data plays a crucial role. Many users while they are using a specific service are usually unaware of the data processing and its consequences. Moreover, the users do not have such information of what is happening to their data and that the control they have to their own data is limited. Lastly, even the penalties for infringements of legal protection of your data take effect after a breach or data misuse has already occured.

"Privacy-enhancing technologies (PETs)" is a term which was introduced for a category of technologies that minimise the processing of personal data. Using PET will decrease the risk of the user's informational privacy. Also, the legal data protection for the data processing of some responsible entities would be more easily fulfilled.

According to [16] the privacy techniques which are used and implemented in the world of internet, the followings can be mentioned:

1. Authentication
2. Attribute based credentials
3. Secure private communications
4. Communications anonymity and pseudonymity
5. Storage privacy
6. Private information Retrieval

The information about privacy techniques deployed in Amazon Web Service and Microsot Azure is limited. Finding the proper information about the *authentication* and the *attribute based credentials* was not an easy task. There are different authentication techniques used by cloud providers as listed below:

1. Multi-factor Authentication
2. Public Key Infrastructure
3. Username and password
4. Single sign-on
5. Mobile Trusted Module
6. Biometric Authentication

2.3.1 Amazon Web Services(AWS)

AWS together with MA are the best-known cloud platforms in the world. In chapter 4, I have described how the application in AWS is deployed. Moreover, in chapter 4 is described how AWS works, how to create and manage the EC2 instances, how to upload the files in a bucket (S3 Storage) and how to connect EC2 instances with S3 Storages. For that reason in this section, an overview of AWS is given.

AWS platform is based on the concept of "Pay-As-You-Go". AWS has a bunch of services which clients can use when it is required, without any long term commitment. It enables clients to use services, such as **computing, database storage, networking and programming models.**

Moreover, Amazon offers a free usage of AWS cloud platforms for its costumers, for a year. After the expiration of a free usage tier, you are required to "Pay-As-You-Go". Amazon as a free usage tier offers the following:

1. Amazon EC2 virtual server to resize computing capacity in the cloud, with 750 hours per month.
2. Amazon S3 storage infrastructure with standard storage of 5 GB facilitating 2.000 Put Requests and 20.000 Get Requests.
3. Amazon DynamoDB with 25 units each of Read and Write capacity and a stoarage of 25GB.
4. AWS IoT, device to cloud connector, which can deliver 250.000 messages each month.
5. Amazon EC2 Container.

Regions and Availability zones AWS is made up of regions which are a grouping of independetly seperated data centers in a specific region known as "Availability zones". It is important to note that different regions have different prices for their services. There are 6 AWS regions as listed below:

1. Europe(Dublin)
2. US East(Northern Virginia)
3. US West(Northern California)
4. Asia Pacific(Tokyo)
5. Asia Pacific(Singapore)
6. GovCloud(West Coast)

Availability zones work together in a region to make up a collection of our AWS resources. Availability of regions allows the architect to design applications to conform to specific regulations and laws for a particular part of the world. Viewing a region in the console means that you will only view resources in one region at a time. There are some AWS services which will work globally, not only in a particular region. One example, is the users created in IAM.

Use cases AWS can be used in several occasions.

Manufacturing Organization can use AWS to extend its business through quality production while leaving IT Management to AWS.

Media Company can use AWS in order to provide different types of content, such as audio files and videos.

Architecture consulting company can use AWS to get high-compute of construction prototypes.

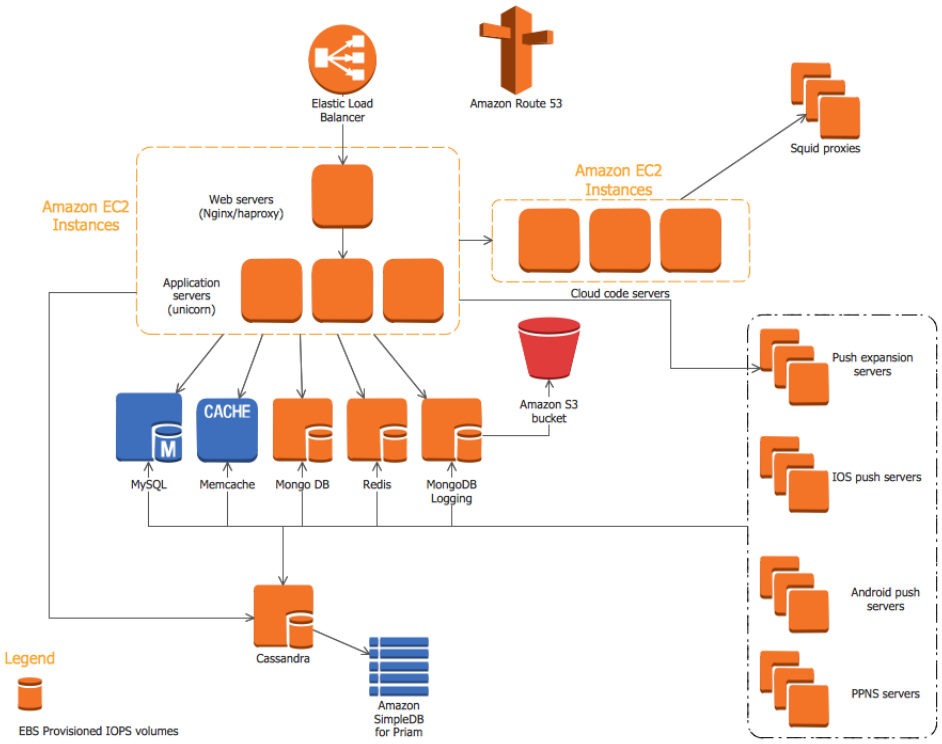


Figure 2.4: AWS architecture. Image retrieved from [14]

Accessing AWS can be done in 5 different ways in order to create and manage our applications.

AWS Command Line Interface which refers to a command line tool. This tool operates through commands for managing and automating our AWS services.

AWS Management Console which refers to a web interface.

Command Line Tools which operates through commands in order to manage our individual products.

Query API refers to low-level API accessible through HTTP requests.

AWS Software Development Kits(SDKs) refers to application programing interface or API specific to our programming platform.

AWS services are grouped together by ”type”. AWS are grouped in the following categories:

1. Storage
2. Database Services

3. Compute and Networking
4. App Services
5. Management Services
6. Deployment Services
7. Analytics

A properly architected application will make use of multiple types of services as it is shown in Fig 1.4.

Privacy and Security in AWS

Security

AWS provides data security by using network architecture which helps us meet security related objectives, such as: Alertness, Visibility, Auditability, Manageability. Information related to the security, AWS delivers to clients by using different mediums (e.g, papers, certifications, reports, Third-party Attestations). This kind of information plays a crucial role by giving the clients the neccessary information about AWS security controls and how the third-party provider would validate these controls.

When it comes to the security, AWS offers its customers security benefits by ensuring them with a robust safety mechanism. The security managers which are hired by AWS, review and verify the components on the daily basis. AWS offers to its clients an online tool called Trusted Advisor which identifies security gaps and fills them. AWS uses different security tools in order to secure clients data. The key features of these security tools are the followings:

- Provide data encryption and build-in network firewalls
- By using tools as Amazon Inspectors it evaluates applications for weakness or deviations.
- By using AWS Identity and Access Management, and AWS Mulit-Factor Authentication it defines hardware-based authenticators and user account permissions.
- Maintain and monitor logs of access and changes at the customer's AWS environment.

AWS implements a share responsible model to ensure the security of data and its products and services. It provides multiple data protection services, such as: encryption, security groups and multi-factor authentication capabilities.

Datacenters are a crucial component of AWS. It is very important to keep them physically secured. To achieve that, AWS undertakes the following key measures:

- Implements two factor authentication
- Allows only individuals with approved and authorized access
- Provides non-stop monitoring and auditing of physical access controls
- Deploys training security guards

AWS uses monitoring tools in order to provide security for all products. These tools monitor: Applications, usage of network and server, port scanning activities and unauthorized attempts. Moreover, AWS uses some tools in order to protect the entire environment. The key measures implemented by this tools include, SSL for encrypted transmission over HTTP, only users with cryptographic keys and certificates can access an AWS API and offer data encryption of files and objects stored using AWS services (e.g, Amazon RedShift, Amazon Glacier, and others).

Privacy

In [39] is described in details how protection mechanisms works in AWS. Some of these mechanisms are listed below:

- AWS Account security features
- AWS Overall authentication

1. **AWS Account Security Features** AWS provides a variety of tools and features that you can use to keep your resources and AWS account safe from unauthorized use. This includes credentials for access control, the creation of separate IAM user accounts, HTTPS endpoints for encrypted data transmission, user activity logging for security monitoring, and Trusted Advisor security checks [39]. The table below highlights various AWS credentials.

2. **AWS Overall authentication** The authentication is a process which gives individual access to the system based on credetentials the client provides to authenticate. In AWS platform authentication process and steps to follow are described in details on [40].

   The corresponding steps of authentication process for the clients are given below:

   - If you do not have an Amazon account you must create one or if you have one just log in (note: This is just a regular Amazon.com login)
   - To have access and to use your product you have to sign up
   - Download your product
   - Downloading the product requires an activation key
   - Activate your product by using this activation key

Credential Type	Use	Description
Access keys	Digitally signed requests to AWS APIs(using CLI,AWS SDK)	Includes a secret access key and an access key ID. You use these keys to digitally sign programmatic requests that you make to AWS.
Key pairs	(a) CloudFront signed URLs (b) SSH login to EC2 instances	A key pair(AWS use 1024 bit SSH-2 RSA keys)is required to connect to an EC2 instance launched from a public AMI. You can upload your own key pair or you can have a key pair generated automatically for you when you launch the instance
Passwords	IAM user account or AWS root account (is a web service that assists you securely control access to AWS resources for your users) login to the AWS Management Console	A string of characters provided for logging into your AWS account or IAM account. AWS passwords must be a minimum of 6 characters and may be up to 128 characters
MFA(multi factor authentication)	IAM user account or AWS root account or login to the AWS Management Console	It is required in additon to your password in order to log in into your AWS account or IAM account. It is 6 characters long
X.509 certificates	(a) SSL server certificates for HTTPS (b) Digitally signed SOAP requests to AWS APIs	They are used only for the purpose of signing SOAP-based requests. You can upload your own certificate by using Security Credentials page

Table 2.1: Credentials types and their use in AWS platform

- The client downloads and installs your product
- During installation, or just after installation, the product, as it is shown in Fig 1.5, goes through an activation process. After the product obtains the activation key from the client, your product sends an inquiry to the license service to activate itself and obtains an access key ID , secret access key and user token for the customer. Your inquiry involves the product token for your product and the client's activation key. The product appropriately saves the credentials it has received, so the client can use the product seamlessly in the future.

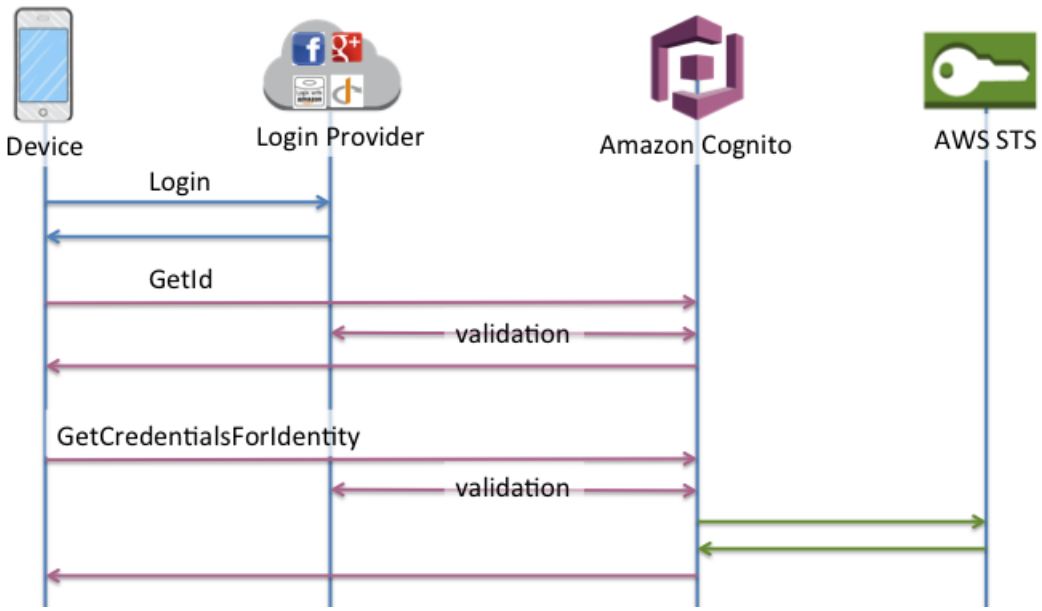


Figure 2.5: Authentication mechanism in AWS platform. Image taken on [12]

Access key ID is a string of 20 characters and Secret Access key is a string of 40 characters.

- Later, when the client uses the product, it makes an Amazon S3 REST request on behalf of the client. In the process, the product reclaims and uses the access key ID, secret access key, user token and the product token for the product. Amazon S3 requests that use DevPay must be REST requests. SOAP requests are not supported for DevPay.

There are several other privacy techniques which AWS platform has implemented in order to protect the privacy of their costumers. However, despite the attempts Amazon has done to make its environment entirely safe, is a fact that no system which can be accessed over the internet is entirely safe. For that reason, security and privacy of each cloud platform, including AWS, is counted more as a disadvantage rather than an advantage.

2.4 Summary

In this chapter, is done an overview of what cloud computing is. Cloud's model services and its deployment models are described. Later, the advantages and

disadvantages of using cloud computing are discussed. The main conclusion from this chapter is that cloud computing is becoming very popular nowadays and it is affordable in terms of money. Also easily can be pointed out that the main concern is security and privacy of cloud platforms. In the last section of this chapter, are described some of the privacy techniques, such as authentication in real word scenarios, like Windows Azure and Amazon Web Service cloud platforms. Since there are a lot of highly secured authentication protocols implemented and deployed to the existing cloud platforms, I decided to dedicate my work to another privacy technique, such as private information retrieval. According to the searching over the internet, there is no single cloud system which has implemented and deployed PIR protocol and that gives me more motivation to work toward this topic.

Chapter 3

Data Beyond Privacy

3.1 Introduction

With the advance of technology, the internet has become a widespread technological tool used for various purposes. It has been reported that the number of internet users has increased from 738 million in 2000 to 3.2 billion today. It means that the internet traffic is very dense and each day it continues to expand. By internet traffic we understand data which flows over the internet. Every internet user in his daily basis uploads and downloads data from the internet. Data is our most valuable asset. It is so important to keep our data safe over the internet, especially in the case of sensitive data. In the last decade there have been a lot of cases with data breach. In terms of money it costs a lot.

According to this report [24], in 2014 happened the largest hack of all the time. Yahoo confirmed that about 500 million user account credentials have been stolen. The stolen data include email addresses, names, telephone numbers, hashed passwords and birthdays. It cost Yahoo a lot of money. According to some reports, this breach could have implications for the 4.8 billion sale of Yahoo to Verizon. This case was even larger than the case of Myspace breach of 427 million password and 360 million user accounts. There are a dozen of other cases in which data breach has occurred.

As previously stated, no system which can be accessed by internet, is 100 % safe. Cloud computing, because of the advantages it offers to the organisations, is becoming each day more popular. However, a lot of organisations are afraid to trust their data to a third-party provider. Despite of the high-level security, privacy algorithms cloud platforms have deployed in their cloud environment, the cloud platforms are not entirely a safe environment.

In this chapter are discussed privacy aspects which go beyond data privacy. Even if the shared information during the transmission is encrypted, other sensitive information may be revealed to an unauthorized user. Such information contains the identities of sender and receiver, how often the communication be-

tween two parties is done, when the communication starts and the message size. In many situations, revealing any of these information may be problematic, even if the content of the data is encrypted.

In a storage system such side channels exist. A third-party provider can reveal relevant information. For instance, the information may contain who accesses the data, who has access to the data, from where the data is accessed and how often data is accessed. Moreover, the third-party provider may reveal how much data are stored. By using these side channels, third-party provider can learn the access history. The access history may reveal user's habits and preferences. Possessing information about the access frequency of files stored in a cloud platform may lead to the individual data revealing.

In this chapter the focus is mainly on the privacy algorithms which help us hide information to the third-party provider about which data is accessed, how often data is accessed and what action the customer is performing. One of these privacy algorithms is *Private Information Retrieval*. For readers who are interested in this topic and want to know more about privacy aspects, we suggest them to look in [57]

3.2 Secret-Sharing Schemes

To undertand the importance of threshold schemes, let's take a simple problem: 6 detectives are working on a secret project. They want to lock up their documents in a safe box. Safe box can only be opened if and only if 3 or more of the detectives are present. What is the minimum number of the detective's combination in order to open the safe box? What is the minimum number for a particular detective to form a group of 3 detectives? The answer for the first question is 20 and for the second question is 10. So, even for a case with small number of detectives, the minimum number of combinations of detectives needed to reveal the secret is large. Therefore, the objective of developing an efficient secret sharing scheme is to find out one mechanism through which you can divide the secret amongst n participants, such that, if t of them combine, they will be able to find the secret. The scheme should be efficient, which means that the number of combinations should be small. It means that the computational over it should be as minimal as possible.

Secret-sharing schemes are a tool used in many cryptographic protocols. It is an important tool for storage systems. It protects the system against destruction and data leakage. In [58], a survey about existance of secret sharing schemes is done. This survey includes the most relevant constructions of secret-sharing schemes, discussion about the known lower bounds on the share size, two results for connecting secret-sharing schemes for a Hamiltonian structure to an open problem in cryptography.

Shamir [48] and Blakley [59], were the first ones that introduced secret-sharing schemes for the threshold case (n,t). In this threshold case, the secret is divided into n parts, giving each participant a unique part. Then, to reconstruct it, at least t parts, or all of them are needed. Shamir's threshold scheme is more

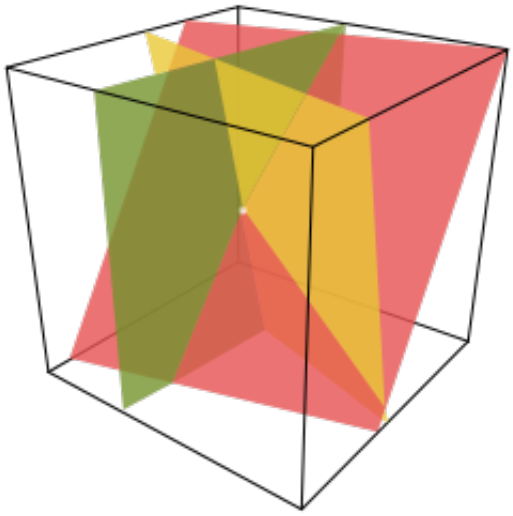


Figure 3.1: Blakley's scheme using three dimensions: A plane represents a share. The point in which three shares intersect, is the secret. Image taken from Wikipedia.

efficient than Blakley's in terms of the share size. In Shamir's scheme the shares are only as large as the original secret, whereas in Blakley's scheme the shares are k times bigger than the original secret. Figure 1.1 visualizes Blakley's secret sharing scheme where 3 non-parallel hyperplanes intersect to reveal the secret.

Shamir's scheme has two major problems: to keep all shares, large storage is required and heavy computational cost is needed. To overcome the first problem, ramp secret sharing schemes have been proposed [66]. To overcome the second problem Ishizu et al. introduced a fast threshold scheme (3,2) in [52]. Fuji et al. generalized Ishizu's scheme for n number of participants (n,2) in [64]. These schemes, in order to create shares and then reconstruct the secret from two or more shares, use only XOR operations. XOR operations enable them fast computational. Secret Sharing schemes for general access structures were presented by Saito, Ito, and Nishizeki [20]. More efficient schemes were introduced in, e.g., [[62], [63]].

3.2.1 Shamir Secret Sharing Threshold Scheme(SSS): Mathematical description

Adi Shamir presented a method for sharing a secret. He suggested taking the secret S and to divide it amongst n participants. Furthermore, the secret S can be easily reconstructed only if we have $t < n$ pieces. The knowledge of t-1 pieces, makes the secret S undetermined. This is called a (n,t) threshold

scheme.

The scheme relies on the idea that you can fit a unique polynomial of degree (t-1) to any set of t points that lie on the polynomial. It takes two points to define a straight line, three points to fully define a quadratic, four points to define a cubic curve, and so on.

In a nutshell, the essential idea of SSS threshold scheme is that 2 points define e polynomial of degree 1, 3 points define a polynomial of degree 2 and 4 points can define a polynomial of degree 3. In general t points define a polynomial of degree t-1.

Figure 1.2 illustrates the Shamir Secret Sharing Scheme. With SSS threshold

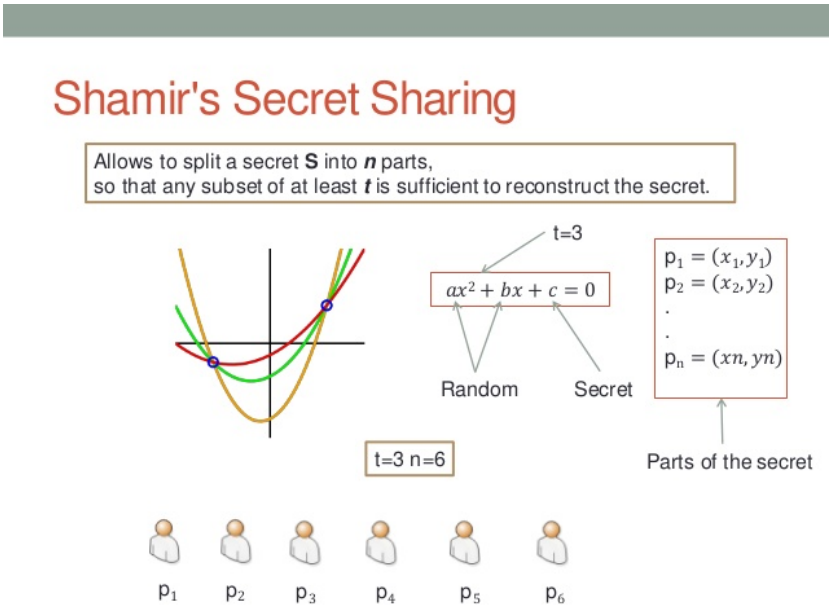


Figure 3.2: Shamir's Secret Sharing Scheme

scheme a solution to two problems is given. The key can be distributed to different users, and each user will keep his/her secret. Moreover, the user by knowing only his/her shares still can not recover the secret. Secondly, the intruder might compromise different user's shares, but the secret generated from SSS threshold scheme will be very difficult to be obtained. SSS is based on polynomial interpolation. It is very important to understand the Lagrange interpolation since it is used in the second phase of SSS, when the secret is recovered.

Lagrange Interpolation

Lagrange polynomials are used for polynomial interpolation. If somebody gives us three data points and if we draw a polynomial through these data

points, it will be a second-order polynomial.

Theorem: Given (k+1) data points as (x_0, y_0), (x_1, y_1), ..., (x_k, y_k) where $x_0 \neq x_1 \neq ... \neq x_k$. Then, k-th order polynomial is given as : $f_k(x)=\sum_{i=0}^{k} L_i(x)f(x_i)$, where $f(x_i)=y_i$ and weighting function $L_i(x) = \prod_{j=0, j\neq i}^{k} \frac{x-x_j}{x_i-x_j} = \frac{x-x_0}{x_i-x_0} \frac{x-x_1}{x_i-x_1} .... \frac{x-x_{i-1}}{x_i-x_{i-1}} \frac{x-x_{i+1}}{x_i-x_{i+1}} .... \frac{x-x_k}{x_i-x_k}$

Example: Define 2−order ploynomial, $f(x)$ when these data points are given: $(x_0, y_0) = (2, 10)$, $(x_1, y_1) = (4, 16)$, $(x_2, y_2) = (5, 20)$. To define a polynomial of degree $k - 1$, it takes k point.

Firstly, let's calculate the weighting function $L_i(x)$.

$L_0 = \frac{x-x_1}{x_0-x_1} \frac{x-x_2}{x_0-x_2} = \frac{1}{6} x^2 - \frac{3}{2} x + \frac{10}{3}$

$L_1 = \frac{x-x_0}{x_1-x_0} \frac{x-x_2}{x_1-x_2} = \frac{-1}{2} x^2 + \frac{7}{2} x - 5$

$L_2 = \frac{x-x_0}{x_2-x_0} \frac{x-x_1}{x_2-x_1} = \frac{1}{3} x^2 - 2 x + \frac{8}{3}$

Therefore

$f(x)=\sum_{i=0}^{2} L_i(x)f(x_i) = \frac{1}{3} x^2 + x + \frac{20}{3}$

How does Shamir Secret Sharing works?

The scheme passes into two steps: sharing the secret and then, recover it.

Example:

First step (Sharing the secret):

Suppose the secret is 3 ($S = 3$). The secret is divided into 6 parts ($n = 6$) where any combination of 3 parts ($k = 3$) is sufficient to recover the secret. The polynomial will be the degree of 2 and the two other coefficients are chosen randomly. Let them be $a = 2$ and $b = 1$.

The polynomial will be on the form: $y = ax^2+bx+c$. After substitution the polynomial will be: $y = 2x^2+x+3$. Figure 3.4 represents the graph of thw polynomial. Six pairs (x, f_x) from the polynomial are constructed : $(1, 6)$, $(2, 13)$, $(3, 24)$, $(4, 39)$, $(5, 58)$, $(6, 81)$. Each participant will receive a single different pair. It is important to note that we do not use here pair $(0, f_0)$ since it will give us the secret.

Second step (Reconstruction):

Now that the shares and the threshold scheme (6,3) are available, the secret can be reconstructed. In order to recover the secret any three combination of 3 pairs are enough. Let it takes the first three shares $(x_0, y_0) = (1, 6)$ $(x_1, y_1) = (2, 13)$ $(x_3, y_3) = (3, 24)$. The *Lagrange basis polynomials* is computed as it is done in the example above. After some calculations based on the formula above the weighting functions are as below:

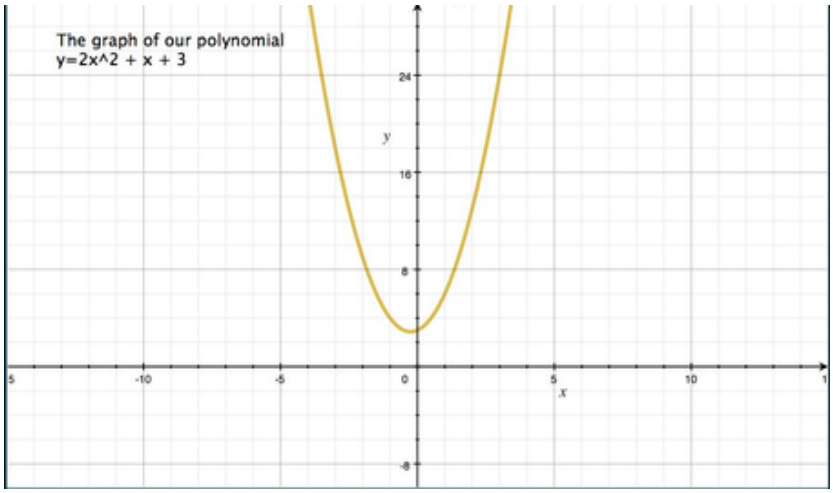


Figure 3.3: Shamir's Secret Sharing Scheme

$L_0 = \frac{1}{2}x^2 - \frac{5}{2}x + 3$

$L_1 = -x^2 + 4x - 3$

$L_2 = \frac{1}{2}x^2 - \frac{3}{2}x + 1$

After applying formula $y=\sum_{i=0}^{k} L_i(x)f(x_i) = 2x^2+x+3$. Recall that the secret is the free coefficient of the polynomial, so S=3 and that is.

It is important to **note** that in order to have an efficient computation of the polynomial in a certain point, Horner's rule is being considered. I will discuss about Horner's rule in Chapter 4. Also, the aim of polynomial interpolation is to find a constant in a source polynomial S=f(0). Since unused constants are calculated using Lagrange interpolation as we have used above, it is not efficient.

Another problem is a security problem. With all pairs you find, you find a lot of information about the secret. Here [18], you can find a detailed explanation of this security problem. To avoid that problem, a finite field arithmetic in a field of size p is used. This prime p should be bigger than the number of participants and any coefficient of the polynomial including the secret. Instead of calculating $(x, f(x))$ calculate $(x, f(x)mod(p))$. In the thesis I decide to work with Galoa Field (256) as it offers us the best performance. Thus, it is important to take a look and explain how GF(256) works.

3.2.2 Verifiable Secret Sharing

Verifiable secret sharing is a protocol in which a dealer or a processor selects and encrypts a "secret" and delivers a "share" of s to each of n participants or processors. The efficiency of VSS schemes is based on two criterias

1. The numbers of bits which must be communicated between participants.

2. The required round numbers of communication.

The aim of verifiable secret sharing schemes (VSS) is to verify the consistency of the shares during both phases, the sharing and the reconstruction [55] . The first version of VSS was presented in [65]. This version assumes that the channels are secure and private. It is unconditionally secure only when $t < \frac{n}{3}$. In [50] the authors assume that broadcast channels exist on the top of private secure channels. It only requires $t \leq \frac{n}{2}$. The first version has a probability of error equal to 0 while the second one has a negligible probability of error.

Feldman [61] and Pedersen [68] have proposed new versions of VSS. The schemes proposed by Feldman and Pedersen are based on homomorphic functions. These functions are hard to invert and is also hard to compute discrete logarithms over prime fields, Z_p.

The protocol proposed by Feldman [61], is a non-interactive scheme VSS and consists of the following phases

1. *Sharing phase* In this phase each participant P_i gets a different non-zero element $e_i \in Z_p$. Let S be the secret in Z_p. This phase consists of three steps.

   - The dealer chooses a random polynomial $f(x)$ of degree k over the finite field Z_p, where $f(0) = S$
   - Dealer computes each share s_i and transmits it to the participant P_i
   - The dealer broadcasts the values g^{ej}, where g is a primitive element in Z_p, for $j = 0...k$ with $f(x) = \sum_{j=0}^{k} e_j x^j$

2. *Detection phase* It consists of two phases.

   - Using this equation $g^{s_i} = \prod_{j=0}^{k} A_j^{eij}$, each participant validates the share. In cases when share fulfilles that equation, the share is valid, otherwise P_i broadcasts an accusation to the dealer.
   - The dealer D is said to be corrupted when the number of accusations broadcasted to the dealer is higher than threshold k.

3. *Reconstruction phase* It consists of three steps:

   - Each participant P_i broadcasts $f(e_i)$
   - Take k+1 broadcast values for which the equation in phase 2 is fulfilled.
   - Output $f(0)$

The protocol proposed by Pedersen [68], is also non-interactive scheme VSS. In his paper he extended Shamir secret sharing protcol with a verification protocol, VP, so any group of k participants, who honestly have accepted their shares in VP, can find the secret s. VP must satisfy the following requirements:

- In the cases when the dealer and P_i both follow VP and the dealer follows the distribution protocol , then P_i accepts with probability 1
- For all subsets S_1 and S_2 of (1,. . . , n) of size k such that all parties (P_i) $i \in s_1$, and (P_i) $i \in s_2$, have accepted their shares in VP the following holds except with negligible probability in q: If s_i is the secret computed by the participants in S_i (for i = 1,2) then $s_1 = s_2$.

If a share is accepted by VP, it is called a correct share. The requirements above guarantee that the dealer almost any time he will try to cheat, he will be caught.

Using secret sharing schemes, there are cases when the dealer can not be trusted. The dealer can transmit corrupted shares from which the participants are not able to recover the secret. Shamir, in his paper did not assume these cases. In contrast, VSS assumes that there are cases when the dealer can not be trusted. This means that VSS is not only useful against adversaries, but can also be vital for detecting errors during transmission of the shares. Moreover, VSS assumes that only authorized groups of participants can learn the secret. The honest participants can learn the secret even if the shares and dealer are corrupted by an adversary.

Before VSS, all previous schemes have been *interactive*. In one interactive scheme, the involvment of participants is needed since they exchange messages in order to prove the correctness of the shares. VSS schemes can be *interactive*, but also can be *non-interactive*. Non-interactive scheme, unlike interactive scheme does not involve the exchange of the messages between participants since the shares can prove its own validity. Moreover, VSS scheme can also be *publicly*. In this scenario, anybody not only participants themselves can verify that participants received the correct shares.

3.3 Finite Fields

I decide that my application uses SSS. As previously stated that a security problem exists in SSS threshold scheme, if a finite field is not used. In the thesis GF(2^8) is used because, mathematical operations on that field can be performed faster than in any other finite field. A Finite Field which can be found in different texts also as a Galois Field, has a finite number of elements. There are 4 basic mathematical operations, which we can apply to elements of the finite field like: addition, subtraction, multiplication and inversion. [41] An additive and multiplicative group is required to have the four basic operations on a field. Group is a set of elements G together with an operation which combines two elements of G.

A Finite Field F is a set of elements with the following properties:

- All elements of field F form an additive group with the following operation "+" and neutral element "0".
- All elements of field F form a multiplicative group with the following operation "x" and neutral element "1".
- When addition and multiplication group operations are both applied, the distributive law takes place : $a(b+c) = ab + bc$.

Order is the number of elements in a field. A finite field with order q only exists if q is a prime power, i.e., $q = (r^p)$, for some positive integers p and prime integers r. Finite Fields are divided in two types: prime (order) fields and extension fields. Prime fields have the $p = 1$. Elements of the field GF(r) can be represented by integers from 0 to $r - 1$. The two operations of the field are modular integer addition and integer multiplication modulo p.

In extension fields we can not apply the addition and multiplication in the same way as addition and multiplication of integer modulo. The elements of extension fields are represented as polynomials. The polynomials have a degree of $(q - 1)$.

The main working principle of addition and subtraction in GF(2^q) is by performing standard polynomial addition and subtraction. We add or subtract equal powers. The coefficient additions or subtractions are done in the underlying field GF(2). This operations are computed according to equations

$$G(x) = F(x) + R(x) = \sum_{i=0}^{q-1} g_i x^i mod 2, \text{ g(i)=f(i)+r(i) mod 2}$$

Always keep in mind that we use modulo 2 subtraction(addition) with the coefficients.

The principle of multiplication is as follows. Firstly, two elements of the finite field which we represent by polynomials, are multiplied by using the polynomial multiplication rule. All the coefficients are part of GF(2). The product of polynomial multiplication is divided by a certain polynomial (irreducible polynomial) and only the remainder after the polynomial division is considered. An irreducible polynomials for the module reduction is needed. Recall that irreducible polynomials are roughly comparable to prime numbers, their only factors are 1 and the polynomial itself.

In a given field GF(2^q) and the corresponding irreducible polynomial p(x), the inverse of (a^-1) of a non zero element, where a $\in$ GF(2^q) is calculated according to:

- Let $A(x)$ & $B(x) \in GF(2)$
- $C(x) = \sum_{i=0}^{q} c_i x^i mod 2, c_i \in GF(2)$
- Multiplication of two elements is performed as: $C(x) \equiv A(x) * B(x) mod P(x)$

3.3.1 Generators in Fields

A really important concept of finite fields are *generators*. Generator is an element in which all the following powers take on all the powers starting from 1 and so on [54]. Let's take as an example the field z(13). Firstly, let's try power of 2 with modul 13

$(2^1) = 2$; $(2^2)\%13 = 4$; $(2^3)\%13 = 8$; $(2^4)\%13 = 3$; $(2^5)\%13 = 6$;
$(2^6)\%13 = 12$; $(2^7)\%13 = 11$; $(2^8)\%13 = 9$; $(2^9)\%13 = 5$;
$(2^9)\%13 = 10$; $(2^9)\%13 = 7$; $(3^6)\%13 = 1$;

As a result, is clear that 2 is a generator. It is obvious that generators are hard to be found. The polynomial x+1, which can be written 0x03, is the simplest generator for Galois Field GF(2^8). Its powers represent all the 255 values of the field, except 0. The table in Figure 1.4 is a table of "anti-logs" or "exponential". This table gives each possible power. E_{rs} is the field element and is given by $(03)^{rs}$. The table in Figure 1.5 is a table of "logarithm". In this table L_{rs} is a field element which fulfills $rs = 03^{L(rs)}$. These tables are created by using a multiplying function. Then after the generation of these "look up" tables, a mathematical operation based on these "look up" tables is performed. For addition and subtraction (simple XOR) is not necessary to use look up tables. You should use these tables only for multiplication and multiplication inverse. For instance, let's multiply $ff * e8$. By using the logarithm table is

Table of ``exponentials'': E(rs) = 03^rs

E(rs)	s															
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**a**	**b**	**c**	**d**	**e**	**f**
0	01	03	05	0f	11	33	55	ff	1a	2e	72	96	a1	f8	13	35
1	5f	e1	38	48	d8	73	95	a4	f7	02	06	0a	1e	22	66	aa
2	e5	34	5c	e4	37	59	eb	26	6a	be	d9	70	90	ab	e6	31
3	53	f5	04	0c	14	3c	44	cc	4f	d1	68	b8	d3	6e	b2	cd
4	4c	d4	67	a9	e0	3b	4d	d7	62	a6	f1	08	18	28	78	88
5	83	9e	b9	d0	6b	bd	dc	7f	81	98	b3	ce	49	db	76	9a
6	b5	c4	57	f9	10	30	50	f0	0b	1d	27	69	bb	d6	61	a3
7	fe	19	2b	7d	87	92	ad	ec	2f	71	93	ae	e9	20	60	a0
8	fb	16	3a	4e	d2	6d	b7	c2	5d	e7	32	56	fa	15	3f	41
9	c3	5e	e2	3d	47	c9	40	c0	5b	ed	2c	74	9c	bf	da	75
a	9f	ba	d5	64	ac	ef	2a	7e	82	9d	bc	df	7a	8e	89	80
b	9b	b6	c1	58	e8	23	65	af	ea	25	6f	b1	c8	43	c5	54
c	fc	1f	21	63	a5	f4	07	09	1b	2d	77	99	b0	cb	46	ca
d	45	cf	4a	de	79	8b	86	91	a8	e3	3e	42	c6	51	f3	0e
e	12	36	5a	ee	29	7b	8d	8c	8f	8a	85	94	a7	f2	0d	17
f	39	4b	dd	7c	84	97	a2	fd	1c	24	6c	b4	c7	52	f6	01

r

Figure 3.4: Table of "exponentials". Image taken on [54]

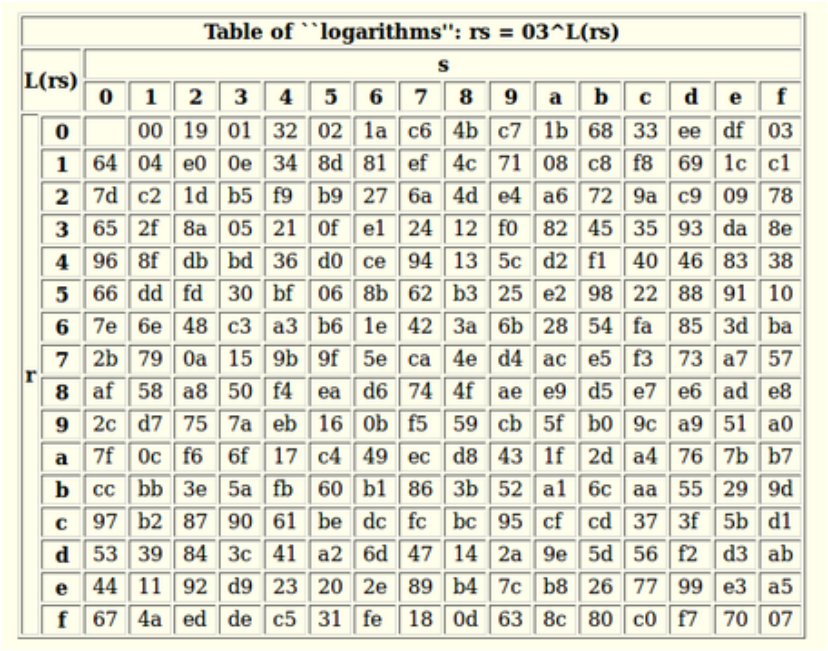


Table of ``logarithms'': rs = 03^L(rs)

L(rs)	s 0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
r 0		00	19	01	32	02	1a	c6	4b	c7	1b	68	33	ee	df	03
1	64	04	e0	0e	34	8d	81	ef	4c	71	08	c8	f8	69	1c	c1
2	7d	c2	1d	b5	f9	b9	27	6a	4d	e4	a6	72	9a	c9	09	78
3	65	2f	8a	05	21	0f	e1	24	12	f0	82	45	35	93	da	8e
4	96	8f	db	bd	36	d0	ce	94	13	5c	d2	f1	40	46	83	38
5	66	dd	fd	30	bf	06	8b	62	b3	25	e2	98	22	88	91	10
6	7e	6e	48	c3	a3	b6	1e	42	3a	6b	28	54	fa	85	3d	ba
7	2b	79	0a	15	9b	9f	5e	ca	4e	d4	ac	e5	f3	73	a7	57
8	af	58	a8	50	f4	ea	d6	74	4f	ae	e9	d5	e7	e6	ad	e8
9	2c	d7	75	7a	eb	16	0b	f5	59	cb	5f	b0	9c	a9	51	a0
a	7f	0c	f6	6f	17	c4	49	ec	d8	43	1f	2d	a4	76	7b	b7
b	cc	bb	3e	5a	fb	60	b1	86	3b	52	a1	6c	aa	55	29	9d
c	97	b2	87	90	61	be	dc	fc	bc	95	cf	cd	37	3f	5b	d1
d	53	39	84	3c	41	a2	6d	47	14	2a	9e	5d	56	f2	d3	ab
e	44	11	92	d9	23	20	2e	89	b4	7c	b8	26	77	99	e3	a5
f	67	4a	ed	de	c5	31	fe	18	0d	63	8c	80	c0	f7	70	07

Figure 3.5: Table of ”logarithms”. Image taken on [54]

obvious that $L_{ff} = 0.7$ and $L_{e8} = b4$. This means that

$$(ff) * (e8) = (03)^{07} * (03)^{b4}$$
$$= (03)^{07+b4} = (03)^{bb}$$

If the sum is bigger than ff just subtract 255. Now, use exponential table to look up $(03)^{bb}$ and to get the result $b1$.

To find the multiplicative inverse of each element field, use again the look up tables. If p is generator 03, then the inverse of p^{rs} is p^{ff-rs}. For instance, if I need to find the inverse of $a4$, I look in the ”logarithmic table” to see that $a4 = p^{17}$, so the inverse of $p^{ff-17} = p^{e8}$, and from exponential table is clear that the result is $8f$.

In chapter 5 a detailed implementation of GF(2^8) is given. Some measurements are done about how fast multiplication and multiplication inverse of the field are. The results are shown at chapter 5. According to the results, is not difficult and is clear to come in conclusion that GF(2^8) is the fastest one. This comes as the result of the usage of ”look up” tables for mathematical operations, such as multiplication and multiplication inverse.

3.4 Private Information Retrieval (PIR)

Private information retrieval is a database technique that uses idea from cryptography to protect the privacy of the users by hiding the queries details from the administrators of the database. The basic idea of PIR is, that we need to protect the identity of the query not the identity of the user. If we want to protect the identity of the user we can use *Tor*.

Basically you can seek a record from the database and database server sends everything back to you. The privacy protection in that case is perfect but communication overhead is huge. This is not practical and for that reason is needed something to make PIR Protocol practical in real world. *Non-triviality* is what is required in order to make it practical. *Non-triviality* refers to, communication cost must be in $O(n)$, where n is the number of bits in the database.

Regarding security, PIR scheme has to provide the following two characteristics.

- Correctness : Every time a client queries a particular block i-th in the database, he receives correctly that particular block.
- Privacy: The database operator does not gain any information about which particular block in the database, the client is quering of.

PIR protocols can be classified according to the following criteria:

1. The privacy they offer to the clients.
2. If data are hosted by a single server or by multiple servers which are not communicating with each other.

According to the privacy, they can be classified as follows:

- Computational PIR (cPIR). Provides privacy against computationally bounded server(s). In that class, the query is encoded in such a way that the server can serve the data and learn nothing about the retrieved data and the queries. Goldberg and Olumofin in [67], have proved that a *cPIR* protocol can outplay the trivial PIR protocol. They here showed that *cPIR* protocol proposed by Gaborit and Melchor [60], is even faster than trivial one when using common network connections. They proved it by using empirical results. Main advantage of cPIR protocol is that the communication cost is reduced by using *recursion*.
- Information-theoritic PIR (itPIR). Provides privacy even against computationally unbounded server(s). Goldberg and Olumofin in [67], here also have proved that an *itPIR* protocol can have better performance than the trivial PIR protocol. In [19] Chor et al. showed that there is no non-trivial itPIR in the case when there is only one single database server.

In [**?**] the authors have combined the positive aspects of *cPIR* and *itPIR* protocol. This protocol is called *hybrid* PIR. It includes the recursive advantage of *cPIR* and the low computation and communication cost of *itPIR*. This *hybrid* PIR has lower costs than PIR protocols of *cPIR* and *itPIR* classes. It is well-suited for databases which can hold large numbers of small data.

According to whether the data is hosted by a single server or multiple non-communication servers, they can be classified as follows:

- Single-Server PIR. A single server hosts the entire database and all the queries are sent to this server.
- Multi-Server PIR. Database is replicated to $l > 1$ servers and each server holds an entire copy of x. For $k \leq l$ servers, the queries are issued concurrently. Over the years different improvements to this simple *l-servers* have been proposed. In [19] a simple (l-1)-private l-server PIR scheme has been proposed. This scheme involves two-server with communication complexity $O(n^{\frac{1}{3}})$, and it is based on XOR queries. This type of query is common and practical since it is deployed on many "real world" databases. The problem with this scheme is, that this scheme is not robust. If any of 2 servers do not respond or just one bit flips, the user will not obtain the correct result. Goldberg in [17] proposed a scheme which is robust and eliminates the above problem. The approach in this paper yields a $t - private\ v - Byzantine - robust\ k - out - of - l$ PIR. This means that the scheme involves l replicated versions of the database, provides query privacy if up to t servers collude and for reconstructing the query result it is sufficient that at least k servers respond, from which at most v are allowed to respond byzantine. This byzantine-robust PIR scheme provides protection for information-theoritic privacy against coalitions of up to all servers. One of the responding servers, hones the result with a factor of 3. Moreover, this protocol provides information-theoritic protection of database contents against collusions of limited number of the servers. Additionally, even if some of the servers return the incorrect information, the user is still able to compute the correct result. To add robustness this scheme also uses Shamir secret sharing threshold scheme. The communication cost is $O(l)$.

The effectiveness of PIR depends on the measurement of the *communication complexity* and *storage overhead*. Communcation complexity, is the total number of bits which are exchanged between the user and the servers. Storage overhead, is the ratio between the total number of bits stored on all the servers and the number of bits in the database.

There are three basic ways how to achieve *non-triviality*, as the followings:

1. Homomorphic cryptography. Homomorphic encryption allows to perform mathematical operations over a ciphertext, and it generates an encrypted result. When this result is decrypted, it matches the result of operations performed over a plaintext. In [46], Kushilevitz and Ostrovsky

. Image taken on [3]

Figure 3.6: Basic idea of Private Information Retrieval

proposed a PIR scheme which uses a cryptographic construction based on homomorphic encryption. This generic construction is based on Damgard-Jurik cyptosystem. The communication overhead here has complexity $O(kd^2\sqrt[d]{n} + kd)$, where n is the database size in bits, k is a security parameter, and $d \in N$. The first practical fully homomorphic encryption (FHE) system was published by Gentry [56]. In [56], he also published an abstract method of how to construct FHE from rings. The key of this scheme is bootstraping the construction. By bootstraping construction, the ciphertext can be updated by homomorphically evaluating the decryption circuit. The new ciphertext has only gone through a circuit as deep as the decryption circuit. Since this paper by Gentry was published, two other FHE schemes have been published. One is published by Smart and Vercauteren [47], which improves the original system proposed by Gentry. Another one is published by van Dijk [51], which is based on the construction of the integers.

2. Trusted hardware. Hardware-based PIR uses trusted hardware which reduce computation and communication complexities. There are not too much PIR protocol based on trusted hardware. The first approach was done by Smith and Safford [23], and it was proposed solely to reduce the communication complexity. For each query, a trusted hardware reads all data from one external database and returns the requested data back to the user. Iliev and Smith in [49] and [53] proposed a PIR scheme based on trusted hardware. Here, for each query, the online process cost is O_1. However, for each k queries, where $k << n$, the database needs to be reshuffled with computation cost $O(nlogn)$. In [21] another PIR scheme based on trusted hardware is proposed. This scheme is information theoretical secured. The communication complexity is $O(logn)$, offline communication cost is $O(n)$ and the online process cost is $O(1)$.

3. Coding theory. In my thesis the problem above is solved by using the third way, *Coding theory*. In the following section the protocol which we

decided to implement will be explained in details.

3.4.1 Private Information Retrieval based on Shamir Secret Sharing

In this section an analytical description and one example of the protocol is given. The protocol is **k-private v-Byzantine-robust t-out-of-n PIR**. It means that if only t of n servers need to respond and even though k servers collude, no information of query will be revealed and v of those t servers are Byzantine. The choice of Finite Field (FF), the set of Shamir indices (I) $\in$ FF have to be made *in advance* of storing data in the database. This protocol allows the content of the database not to be revealed from coalitions of up to τ servers ($\tau-$ **independent PIR**). The communication cost here is O(n) times the size of the block, where n is the number of servers. This scheme has two restrictions in which cases it can not be used, as the followings:

- If the user stores information in database which is different from what client wants to retrieve, or if there is more than one such client.
- In the same time not more than one client can retrieve the data from the database. Multiple clients could not use the same database since they would not use the same FF.

Analytical descritpion

Parameters:

n: total number of servers

k: desired privacy level (number of servers which can collude and learn nothing about the client's query.)

d: size of the database (in bits). The database here is treated as a matrix. Each matrix is rectangular with n-rows and b-columns.

b: size of the blocks (in bits). If all the files are the same size (in bits), then each file is treated as a single block.

w: size of word within a block (each word is 8 bit)

FF: $2^w = 2^8$

I: a set of Shamir indices from FF such that I $\geq$ n

Calculate:

r: n/b blocks. The database is divided into d/b b-bit blocks.

s: b/w words. Each block in a database is divided into b/w w-bit words.

Let's suppose $W_{j1}, ..., W_{js}$ represent the words in a particular block j and $w_{j1}, ..., w_{js}$ represent the Shamir secret sharing computation of the above words. Each server stores a block as a sequence of s elements. All s elements are part of the FF(256). For instance server i stores the block j as a sequence of s

elements $w^i_{j1}, w^i_{j2}, ..., w^i_{js}$ of FF(256). The elements w^i_{jc} are computed using Shamir secret sharing scheme.

For each word W_{jc} in a particular block j a user chooses a random polynomial p_{jc} of degree k. The coefficients of the p_{jc} should be random values of FF(256), except of the free term. The free term should be W_{jc}. Then, he sends p_{jc} (α_i) to server i to store it as its w^i_{jc}, for each $1 \leq i \leq n$. So, the values of w^i_{jc}, for $1 \leq i \leq n$ are k-private n-way Shamir secret shares of W_{jc}.

Client Side (querying a particular block j in the database):

Step 1: Choose a vector of size r (in bits). All elements of the vector, with the exception of the j-th element, have the value 0. The j-th element of the vector has the value 1.

Step 2: Choose n random distinct indices $\alpha_1, ..., \alpha_n$ from I. For the application the values are $\alpha_1 = 1$, $\alpha_2 = 2$, ..., $\alpha_n = n$

Step 3: For each element of the vector choose a random polynomial of degree k. So, in total we have r random polynomials $f_1, f_2, ..., f_r$ each of degree k.

Step 4: Compute $u_i = [f_1(a_i), ..., f_r(a_i)]$ for $1 \leq i \leq n$

Step 5: Send u_i to server number i

Server Side (each honest server):

Step 1: Receive $u_i = [u_{i1}, ..., u_{ir}]$. This is a vector with r elements and all elements $\in$ FF(256)

Step 2: Compute $P_{ic} = \sum_{1 \leq j \leq r} u_{ij} w_{jc}$ for $1 \leq c \leq s$

Step 3: Send back to the client $[P_{i1}, ..., P_{is}]$

Client

Step 1: The client takes $[P_{11}, ..., P_{1s}], ...[P_{n1}, ..., P_{ns}]$ from the n servers. $\beta 1, ..., \beta t$ are the numbers of t servers which respond.

Step 2: When $t \leq k + \tau$, send an error message " not enough servers replied".

Step 3: The minimum number of honest servers is in the range $\sqrt{t(k+\tau)} < h \leq t$

Step 4: Recover the secret using polynom interpolation method. For instance, if the threhold scheme used is (7,3), then to recover the secret 5 servers need to collude.

Why does it work? It is easy to point out that this protocol gives you privacy but you need it also to give you information retrieval. The reason why it gives you information retrieval is because Shamir secret sharing has a nice property which is called *linearity*.

For instance, lets suppose $f(0) = a$ and $m(0) = b$. Then, $(f + m)(0) = a + b$. So it is homomorphic under addition. Moreover, it is also homomorphic under scalar multiplication. For instance, if $g(0) = a \rightarrow c * g(0) = c * a$. To a PIR protocol you have a *vector-matrix multiplication*. This multiplication requires

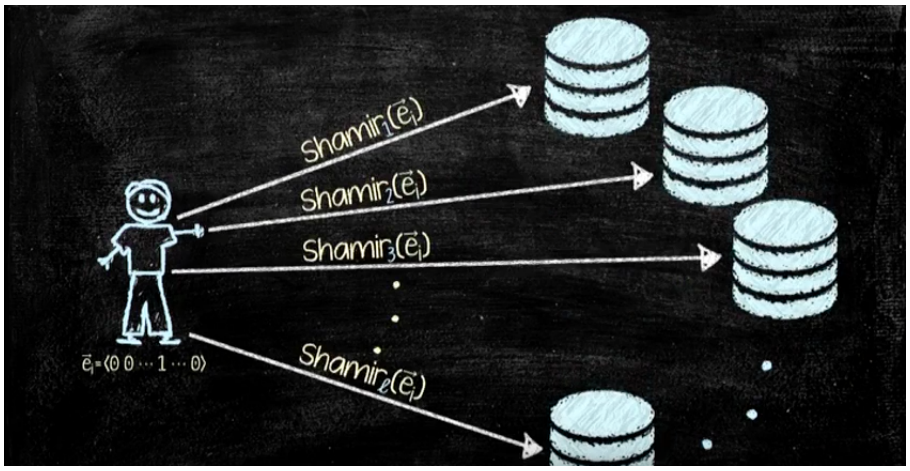


Figure 3.7: PIR based on SSS technique. Each server holds a database with n-rows and b-columns. The values in the matrices are computed using SSS. The vector e keeps the values 0 and 1. The user applies SSS technique over that vector and distributes it to each server

only *scalar multiplication* and *addition* mathematical operations. Also, the reconstruction part requires only *scalar multiplication* and *addition* mathematical operations. Shamir secret sharing scheme is homomorphic under both *scalar multiplication* and *addition* as proved above, so this protcol works perfectly fine. Multiplying the shares of the vector query with the shares of the files is the same as multiplying the real values of vector array which has values 0 and 1 with words W. Below is given an example of the protocol we have implemented.

Example

Setup: Let's suppose the client has a folder with 4 files, each 6 byte, and he wants to query the second file. Let the threshold scheme be $(7, 3)$. All mathematical operations are done over FF(256). The set of Shamir indices (I) is $[\alpha_1 = 1, \alpha_2 = 2, \alpha_3 = 3, \alpha_4 = 4, \alpha_5 = 5, \alpha_6 = 6, \alpha_7 = 7]$. These are the points the polynomials will be evaluated. Each file is treated as a single block and the size of each word is 1 byte. Let's suppose the client's files are as the followings:

1. file1.txt = [c,r,y,p,t,o]
2. file2.txt = [s,h,a,m,i,r]
3. file3.txt = [s,e,c,r,e,t]
4. file4.txt = [p,r,i,v,a,t]

SSS of the files: He performs SSS for each word of the blocks. For the first block (file1.txt) he creates 6 random polynomials (one per each word) of degree

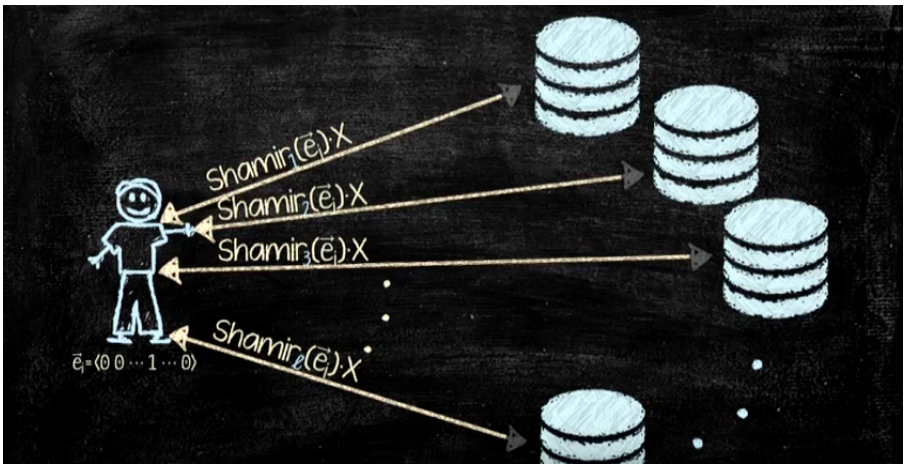


Figure 3.8: PIR based on SSS technique. After vector-matrix multiplication. The client takes back the result of the multiplication from each server and then reconstructs the block which he queries of.

3. The same procedure is for the other blocks. All coefficients of polynomial are randomly chosen and are elements of FF(256) with the exception of the free term which is our secret.

The first word of the first block is *c* and its numerical representation (**ASCII** code) is *99*. This is our secret and at random we obtain two numbers which are part of FF(256) : $a_1 = 118, a_2 = 68$.

Our polynomial to produce secret shares is therefore: $y = 99 + 118x + 68x^2$. The result is ana array V with values V= [81,34,76,143,87,120,52]. After performing the same procedure for other words of the first block, a matrix with 7 rows and 6 columns is created. All values in the matrix are the shared ones. Matrices for all the blocks are given below:

$$M1 = \begin{bmatrix} 81 & -124 & -74 & -116 & -66 & 107 \\ 34 & 95 & 15 & 42 & 122 & 7 \\ 76 & -30 & -41 & -67 & -120 & 38 \\ 143 & -105 & 104 & -83 & 82 & 74 \\ 87 & -2 & -35 & 102 & 69 & -20 \\ 120 & -75 & -94 & 61 & 42 & -65 \\ 52 & 45 & 57 & 6 & 74 & 66 \end{bmatrix} \quad M2 = \begin{bmatrix} 30 & -40 & -75 & -31 & -116 & 74 \\ 117 & 33 & -62 & 31 & 2 & -88 \\ -95 & 23 & -41 & 20 & -44 & 98 \\ -87 & -66 & 122 & -53 & 15 & 24 \\ -76 & -21 & 54 & -6 & 39 & 120 \\ -38 & 123 & -45 & 115 & -37 & 122 \\ 21 & 34 & 57 & 90 & 100 & 39 \end{bmatrix}$$

$$M3 = \begin{bmatrix} -71 & 127 & -75 & 81 & -101 & 93 \\ 34 & -126 & -59 & 15 & 72 & -24 \\ -118 & -120 & 97 & 38 & -49 & -51 \\ 120 & -31 & -21 & 101 & 111 & -10 \\ 42 & -106 & -39 & 55 & 120 & -60 \\ 84 & 27 & 59 & 22 & 54 & 121 \\ 37 & 111 & 101 & 16 & 4 & 12 \end{bmatrix} \quad M4 = \begin{bmatrix} -89 & 97 & 230 & -12 & 35 & -27 \\ -36 & 103 & -55 & 11 & -91 & 30 \\ -73 & -43 & 11 & 102 & -72 & -38 \\ 61 & 90 & 17 & -16 & -69 & -36 \\ 24 & 13 & 116 & -9 & -114 & -101 \\ 127 & 26 & 17 & 83 & 88 & 61 \\ 211 & 3 & 57 & 166 & 147 & 250 \end{bmatrix}$$

Then, take the first rows from each matrix above and store them in the first server, the second rows from each matrix and store them in the second server and the same for all the others. The databases stored in each server are shown below:

$$S1 = \begin{bmatrix} 81 & -124 & -74 & -116 & -66 & 107 \\ 30 & -40 & -75 & -31 & -116 & 74 \\ -71 & 127 & -75 & 81 & -101 & 93 \\ -89 & 97 & 230 & -12 & 35 & -27 \end{bmatrix} \quad S2 = \begin{bmatrix} 34 & 95 & 15 & 42 & 122 & 7 \\ 117 & 33 & -62 & 31 & 2 & -88 \\ 34 & -126 & -59 & 15 & 72 & -24 \\ -36 & 103 & -55 & 11 & -91 & 30 \end{bmatrix}$$

$$S3 = \begin{bmatrix} 76 & -30 & -41 & -67 & -120 & 38 \\ -95 & 23 & -41 & 20 & -44 & 98 \\ -118 & -120 & 97 & 38 & -49 & -51 \\ -73 & -43 & 11 & 102 & -72 & -38 \end{bmatrix} \quad S4 = \begin{bmatrix} 143 & -105 & 104 & -83 & 82 & 74 \\ -87 & -66 & 122 & -53 & 15 & 24 \\ 120 & -31 & -21 & 101 & 111 & -10 \\ 61 & 90 & 17 & -16 & -69 & -36 \end{bmatrix}$$

$$S5 = \begin{bmatrix} 87 & -2 & -35 & 102 & 69 & -20 \\ -76 & -21 & 54 & -6 & 39 & 120 \\ 42 & -106 & -39 & 55 & 120 & -60 \\ 24 & 13 & 116 & -9 & -114 & -101 \end{bmatrix} \quad S6 = \begin{bmatrix} 120 & -75 & -94 & 61 & 42 & -65 \\ -38 & 123 & -45 & 115 & -37 & 122 \\ 84 & 27 & 59 & 22 & 54 & 121 \\ 127 & 26 & 17 & 83 & 88 & 61 \end{bmatrix}$$

$$S7 = \begin{bmatrix} 52 & 45 & 57 & 6 & 74 & 66 \\ 21 & 34 & 57 & 90 & 100 & 39 \\ 37 & 111 & 101 & 16 & 4 & 12 \\ 211 & 3 & 57 & 166 & 147 & 250 \end{bmatrix}$$

Client Side (querying second file)

Now after all shares of the files are stored in each server, the clients wants to query the second file while letting the administator of the storage system know nothing about it.

When the client queries the second file, an array with values [0,1,0,0] is created. Then, the procedure above is repeated. 4 random polynomials of degree 3 are created. Each polynomial has distinct random values for their coefficents

with the exception of free terms which are the secrets. Afer performing SSS, a matrix as below is created.

$$M = \begin{bmatrix} 82 & 105 & 7 & 65 \\ 195 & 73 & 57 & 90 \\ 137 & 42 & 101 & 168 \\ 210 & 198 & 218 & 48 \\ 211 & 30 & 90 & 200 \\ 89 & 109 & 100 & 250 \\ 19 & 240 & 177 & 12 \end{bmatrix}$$

Then, the first row of that matrix is sent to the first server, the second row to the second server and so on.

Server Side

The servers received those arrays with share values. Then, each server performs a vector-matrix multiplication. The multiplication here is a mathematical operation performed over FF(256).

Client

The servers send back to the client the vectors as below.

$$P1 = \begin{bmatrix} 111 & 240 & 4 & 120 & 6 & 187 \end{bmatrix} \quad P2 = \begin{bmatrix} 109 & 5 & 26 & 100 & 209 & 3 \end{bmatrix}$$

$$P3 = \begin{bmatrix} 46 & 12 & 111 & 79 & 78 & 83 \end{bmatrix} \quad P4 = \begin{bmatrix} 13 & 52 & 206 & 183 & 245 & 100 \end{bmatrix}$$

$$P5 = \begin{bmatrix} 50 & 241 & 52 & 200 & 96 & 40 \end{bmatrix} \quad P6 = \begin{bmatrix} 2 & 35 & 224 & 161 & 98 & 137 \end{bmatrix}$$

$$P7 = \begin{bmatrix} 167 & 8 & 198 & 173 & 234 & 2 \end{bmatrix}$$

The client takes them and then reconstructs the secret. To reconstruct it 5 out of 7, servers need to collude. The secret taken back is the second file (file2.txt).

What can be done more

The implementation of this protocol can be extended in order to query q blocks with the cost of one block. Using basic way to query q blocks, a matrix with q rows as below is created. Instead of performing a vector-matrix multiplication a matrix-matrix multiplication is performed.

$$Q = \begin{bmatrix} 0 & 0 & 1 & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 1 & . & . & . & 0 \\ 0 & 1 & 0 & 0 & . & . & . & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & . & . & 1 \end{bmatrix}$$

To avoid that, use *Ramp scheme*. Let's suppose there is a polynomial like : $r_2x^2 + r_1x + S_0$. Using ramp scheme the order of polynomial is increased by one and the two lowest coefficents are two different secrets. The polynomial will be $r_3x^3 + r_2x^2 + S_1x + S_0$. Two secrets can be shared now in the same secret share polynomial.

If q blocks are being queried a vector of polynomials $V = [f_1(x), f_2(x), f_3(x), ...f_r(x)]$ is created. When $f_1(x)$ is evaluated the result is the first column of the matrix Q, the evaluation of second polynomial gives us the second column of matrix Q, and so on.

Results

The table below holds the results for computation cost, storage cost, upload and download for our protocol and for the extended version of it.

Performance report		
	PIR Protocol based on SSS	Extended version
Computation cost	qrs	rs
Storage cost	rs	rs
Upload	qr	r
Download	qs	s

r- the number of rows in database
s- s-word block of data stored in each row of the databse
q-number of blocks we query

3.5 Summary

An overview of existing secret sharing scheme is given. I choose to use Shamir secret sharing as a threshold scheme. An overview of galoa field is given. All basic mathematical operations are performed over galoa field (256) as it offers the maximum speed. All other galoa fields perform mathematical operations slower than this one. The main reason is using look-up tables to perfrom multiplication and inverse multiplication. Later the existing PIR protocol are described. An analytical description of the protocol is given. Then it is extended in order to query q blocks with the cost of one block. To achieve it use rampified scheme. In the end a performance report of the protocol and the extended version is given.

Chapter 4

Deployment of the PIR application in Amazon Web Service(AWS)

4.1 Introduction

In this chapter, the way how the application is deployed in Amazon Web Service is shown. The application is divided into two different activities; the first activity runs in a local computer while the other one runs in Amazon Web Service.

The first activity is an *one-time activity* and the only algorithm used is *Shamir secret sharing*. This algorithm, which is described in details in chapter 3, is used to generate the shares of the files. The second activity runs as many times as we want to query back one file out of m-files and both of the algorithms described in chapter 3 are used. *Shamir secret sharing* algorithm is used to generate the shares of vector query and *Private information retrieval* algorithm is used to calculate the vector-matrix multiplication. Then *Lagrange interpolation* is used to take back the file which is queried by the client.

This chapter has two main sections. In the first one it is given a detailed description of the Amazon Web Service's infrastructure. Also the functionalities of some of the building blocks of AWS are described. The focus is more in *Storage* and *Compute* blocks.

Storage block includes:

- Amazon S3
- Amazon Glacier
- Amazon Snowball

Compute block includes:

- Amazon EC2
- Elastic BeanStalk
- Lambda

The application uses as a compute block 'EC2 instance', and as a storage block 'Amazon S3'.

In the second part, is shown the way this architecture is built and the steps for running the application. Another reason to deploy the application in AWS is, AWS supports HTTP methods. HTTP methods used in the application are *GET* and *POST*. To upload the shares generated from one-time activity two different ways are followed. The first way is using *S3 storages* and connecting them with *EC2 instances.* The second way is using *FileZilla* which is a free FTP solution where both a server and a client are available.

In this chapter, I describe the way how I have built a trusted cloud environment. The threshold scheme used in the thesis is (6,3). The servers in Amazon scenario are EC2 instances. In total are used 7 EC2 instances, where one of them is the client. The client is used only to make a connection with other EC2 instances. After the connection is done the algorithms deployed in these EC2 instance run and the result is sent back at the client EC2 instance. For more details please see the section 3 of this chapter.

4.2 Amazon Infrastructure

Amazon Web Service provides a wide variety of technology services. AWS is a cloud platform with fully-functional, flexible technology infrastructure. To be fully-functional there are some requirements cloud providers need to meet. AWS meets all these requirements and since 2006 it has been the main cloud platform used by different companies. The requirements are listed below:

1. Computation
2. Global Infrastructure
3. Storage
4. Security
5. Application Services
6. Database
7. Content Distribution
8. Development
9. Big Data
10. Deployment and Management

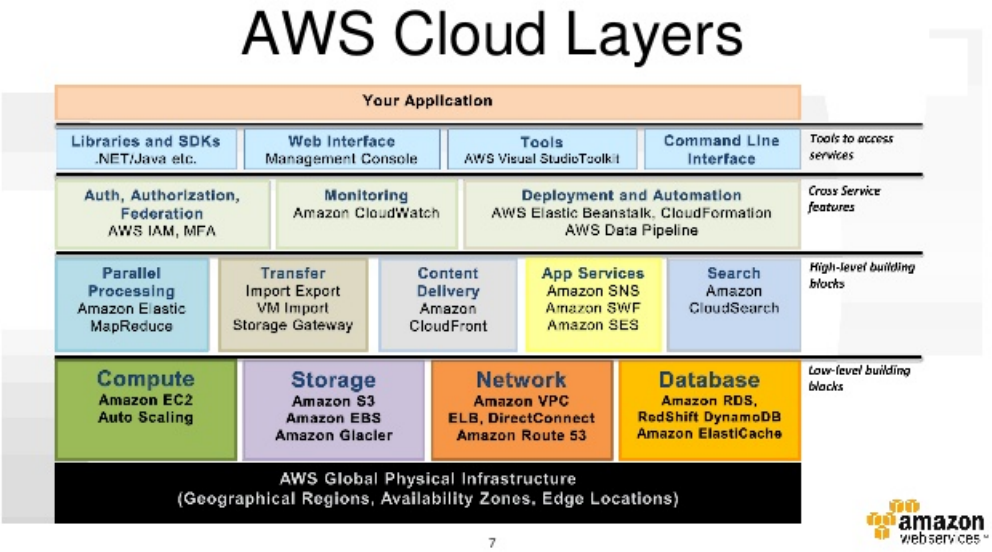


Figure 4.1: AWS architecture. Image retrieved from `http://www.slideshare.net/AmazonWebServices/aws-webcast-29364319`

11. Support
12. Integration with Existing Infrastructure

For this thesis the most important requirements are: security, computation and storage. Security topic is covered in chapter 2. Now let's talk about the two others.

Note: If you want to know more about the other requirements please go to the link `https://aws.amazon.com/choosing-a-cloud-platform/`

Computing block:

Amazon EC2: Amazon elastic compute cloud provides resizable computing capacity in AWS. Using EC2 eliminates the need to invest money in computing hardware up front. In that way money are saved and also it allows applications to be developed and deployed faster. It is a virtual server. It is used to launch a virtual machine and then to configure all networking and security settings. One benefit of using Amazon EC2 is that it can be scaled up automatically and in order to handle extra traffic, it adds capacity. Amazon EC2 is a virtual environment, and Amazon EC2 environments are called instances. There are different types of instances. They can be classified according to the combinations of storage size, CPU power, networking capacity and amount of power. Amazon EC2 instance ranges from small instances which perform small tasks to large instance

with high performance to perform tasks like data warehousing. Moreover, there are even instances with high graphic processing capabilities which can perform tasks like 3D rendering. Based on compute power it is possible to mix and match different types of EC2 instances. Amazon Machine Images (AMI) is provided by Amazon, and it is a pre-configured template for our instances. It can include an operating system like Linux or Windows. They also can include a wide range of components, like preinstalled software packages and operating system.

To create, deploy and terminate an EC2 instance firstly you have to log in to the *AWS management console*. Now click on *Amazon EC2 icon*. To launch the EC2 instance the following steps need to be taken:

1. To begin the process click "Launch Instance".
2. Choose an AMI and click "Select" next to the AMI you want to pick. You can choose Windows, Linux, CentOS etc operating system.
3. Choose an instance type.
4. Configure instance details. In this step access, monitoring, network settings, and other options are added.
5. Add storage. In this step is possible to change the drive size or add more virtual hard drives
6. Tag instance.
7. Configure security groups. It accepts the default security group as it allows SSH from anywhere to this instance.
8. Click on "Review and Launch". By clicking review you can see all your settings for your EC2 instance.

The customers can upload their *Public Key* or create their own *Key Pair* in AWS Management console. To state their access to the private key, they should check the *acknowledgement checkbox*. The role of Key Pairs is to control access to the instance after it is created otherwise will not be able to access the instance. Now after this process is done the instance's state has been changed to running. It is now ready for use. To find the *Public DNS*, check to the box next to the instance. DNS is used to access the system in the same way as it was in data center.

In this thesis 7 EC2 instances are created and launched in that way. The region chosen to deploy the application is Ireland (Dublin) and the operating system used is Linux. While choosing the region three important things should be kept in mind:

- Minimize costs
- Address regulatory requirements

- Optimize latency. If the end users which access the applications are located in Europe they can access the application faster than end users located in other continents. This happens because accessing the application close with the region you have chosen on AWS, reduces the latency significantly.

Storage block:

Amazon S3: Amazon simple storage system is highly scalable, safe and secure object storage in the cloud. It is used to store and retrieve any amount of data, at any time, from anywhere on the web. Using S3 storage makes it easy to store as much data as it is required and it can be accessed it whenever is needed to. One major benefit of using S3 storage is that, it avoids the companies buying hardware and paying for storage that is not being used. You pay only for the storage and bandwith you use. Most of the companies use S3 storage for media hosting, hosting high-traffic websites, for backup, or for software delivery. It can alse be used to store sensitive information and back up our informtation off-site. It also supports versioning, so is possible to go back to any previous version when is required to. It is designed with 99.99% durability. It is very reliable. Moreover it is very easy to use S3 storage. Is possible to programmatically store, retrieve and manage data using REST or SOAP web services. Here, I access S3 storage using REST services. Other ways to access it, is by using command line interface or AWS management console.

Amazon S3 stores data as "objects" and these objects are stored within folders that are called "buckets". Buckets are the container for the objects. Is possible to have more than one bucket. For each bucket the access can be controlled; like, who creates, adds, deletes, and lists the objects in that bucket. The first thing is uploading the file needed to store in the bucket. Later permissions on the object are set. It is private until we decide to share it. Below the way how to create bucket, add objects on it and delete the objects is shown.

1. Click on " Amazon S3" icon on AWS management console homepage.
2. Click on "Create Bucket" in order to create a place where to hold the objects. Then keep the default region "Dublin" unchanged and add a bucket name. The bucket name can not be changed later and is visible to the URL which points to the object saved in that bucket. You only pay for what you store in the bucket and not for creating or deleting a bucket.
3. Click on "Add Files" to upload data. The data uploaded for my application are the shares of the files which are generated by the *one-time activity*. It is very easy to open the file or delete it. The only thing you need to do is to right-click on the file in that bucket and then you can take this operation. The same way with the bucket.

Mount S3 bucket to EC2 instance

To get the shares from S3 bucket and upload them in EC2 instance I have used Java code. The sample code which uploads the shares to EC2 instance is shown below:

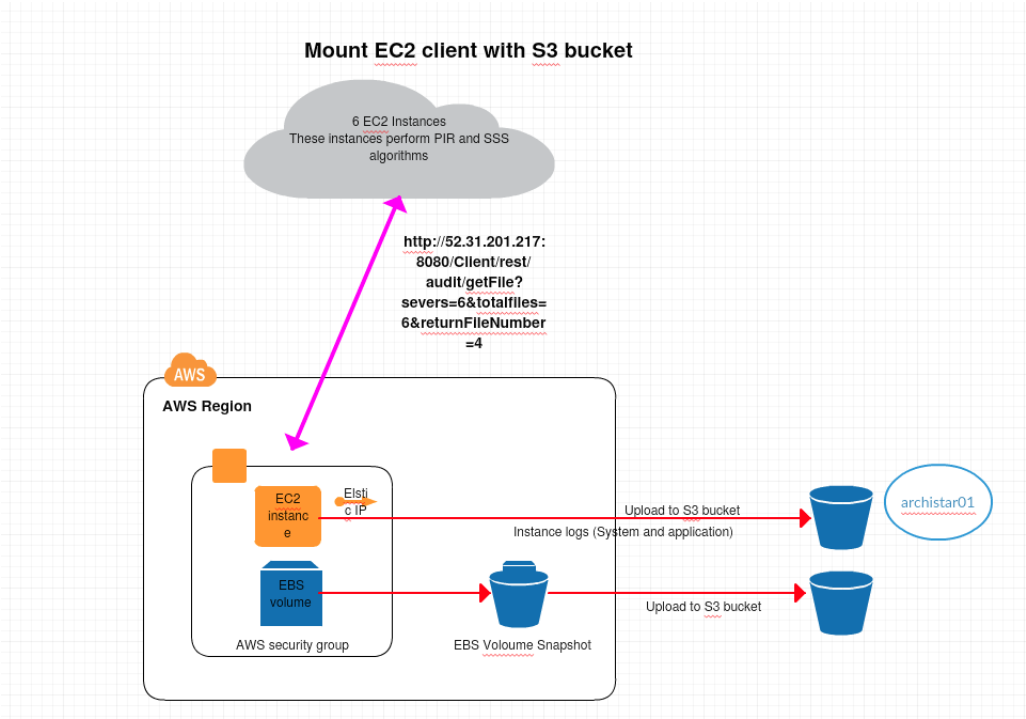


Figure 4.2: EC2 client sending back the file taken by other EC2 instances to our S3 bucket

```
System.out.println("getting share from s3 location");

BasicAWSCredentials awsCreds = new
BasicAWSCredentials("AKIAJR2FJDU2UYMZQD4A",
"DFR0leDYgeBS6Vf0O04oyUHwF6qTRh9d49Pwt+dU");

AmazonS3 s3client =new AmazonS3Client(awsCreds);

S3Object s3object=s3client.getObject(new
    GetObjectRequest(bucketName,"shares.txt"));

  BufferedReader br = new BufferedReader(new
InputStreamReader(s3object.getObjectContent()));
```

After uploading the shares to EC2 instances, the algorithms deployed in EC2 instances will run and then the result is sent back to the EC2 client. The EC2 client will then send the required file back to one S3 bucket named

"archistar01", and from here the queried file can be downloaded (more on this in the next section). The scenario above is shown in the figure 4.2

4.3 Trusted cloud environment architecture

In this section the trusted cloud environment I have built to run the application is shown. As previously stated in total there are 7 EC2 instances. One of them is the client. The application hosted in AWS can be accessed using URL typing in any type of web browser. The URL used has the IP of the client's EC2 instance. After typing the URL in the web browser and run it the client communicates with other EC2 instances and after these, EC2 instances run the algorithms deployed in them, and the result is sent back to the client. The result in this case is one file.

Now two different scenarios are used in order to take back the desired result. One scenario is the client's communication with bucket "archistar01" and the result is sent to the bucket. From here the file is downloaded to the local machine. In the second scenario, after client receives the result, the file is directly downloaded in the local machine. The steps taken to build this trusted cloud environment are shown below:

Step 1:

The application is split in two parts:

- Client part (a single EC2 instance)
- Server part(6 EC2 instances).

Step 2:

1. Shares from the files are generated in the local computer. This is *one-time activity*. Shares from the files are generating using Shamir Secret Sharing algorithm.
2. Then, I copy each share file to each ec2 instance **/home/ec2-user/PIR/shares** folder using FileZila. In the previous section is showed the way how the shares are uploaded to instances using a communication between S3 storage and EC2 instances.

Step 3:

1. Create a post rest service which takes inputs as *no files* as file to be taken, *server number* as query params, and *shareOfqueries* as post data. The functionality of rest web service is as below:
    - Get file from ec2 instance /home/ec2-user/PIR/shares folder
    - Convert data to integer array
    - Get Private information protocol shares using share and share of queries to reconstruct file

- Send this to client as string

2. Create war file for this project
3. Download tomcat server in each instance
4. Extract apache tomcat zip file in each instance
5. Copy war file to **/home/ec2-user/PIR/apache-tomcat-version number/webapps**
6. Run **./startup.sh(/home/ec2-user/PIR/apache-tomcat-version number/bin)** in each instance
7. Now service will be available in each instance

Step 4:

In URL use as parameters, *servers* which is the total number of the EC2 instances(6 in our case), *totalfiles* which is the number of total files, and *returnFileNumber* which is the file we want to take back

1. Call each service from client
2. Convert string to array integer
3. Recover file using recover method
4. Store file into your local machine

In figure 4.3 the architecture of our trusted cloud environment is shown.

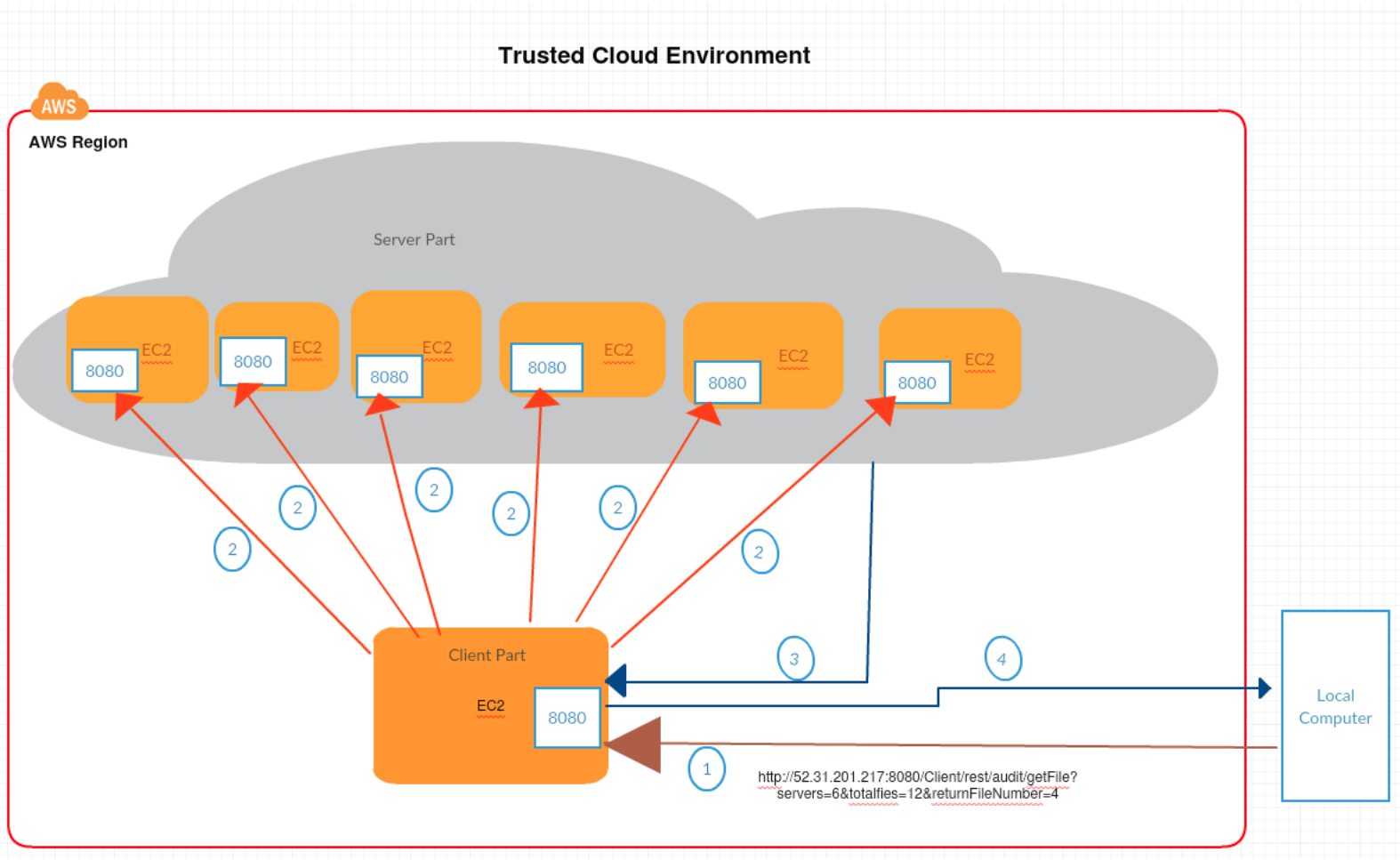


Figure 4.3: The complete scenario of accessing and running the application

4.4 Summary

In this chapter, is shown the way how PIR application in Amazon is deployed. The chosen cloud platform is Amazon web service because of the benefits it provides to. Windows Azure could also be chosen but it is more convenient to deploy the application in AWS. In the first section, the discussion is about the building blocks of the Amazon with more focus in the storage and computing blocks.

In the second section, is shown step by step how the application is deployed in AWS. In the end, the figure 4.3 shows the trusted cloud environment. In order to achieve that, the application is split in two parts. The client part is used only to communicate with server part. All algorithms run in server part and the client takes back the result and then, the result is stored to the local computer from where the application is accessed.

Chapter 5

Implementation Details

In this chapter some implementation details about the application are given. Here is shown which alogrithms are used in order to implement the application. A hard work is done in order to choose the best algorithms which give a desired performance and highly secured random numbers. The methods with the best possible complexity are implemented.

The application has two parts. The first part, is used only to generate the shares of the files. In this part the only algorithm running is Shamir secret sharing. The second part, is a client-server application. The algorithms running here are, Shamir secret sharing and Private information retrieval.

This chapter contains three sections. In the first section, some implementation details about Shamir secret sharing algorithm are given. In the second section, implementation details about the connection between the client and the server are given. Moreover implementation details about Private information retrieval protocol are given. In the end, measurements performance for the most important methods of the application are published.

5.1 Shamir secret sharing implementation details

In order to implement and run SSS some important questions need to be solved. In which finite field mathematical operations will perform on? How to generate random coefficients of the polynomial in order that these coefficients be as random as possible? It turns out that the finite field which gives the best possible speed performance is finite field 256. The reason behind this is, that look-up tables can be used in order to perform multiplication and inverse multiplication operations.

For random numbers generation, since there are different algorithms which can be implemented in order to produce random numbers, a design pattern called *Strategy Pattern* is used. Using that design pattern different algorithms

can be implemented in different classes and then in the methods which calculates shares of the files is possible to put as its argument any type of these algorithms.

Finite Field 256:

Below is represented the code which describes how the finite field is created, and some methods which do basic mathematical operations.

```
package galoisField;
public class FastFiniteField256 implements galoa{

 //exponential table a vector with 256 elements
 private static int[] Exp = new int[256];

 //logarithm table vector with 256 elements
 private static int[] Log= new int[256];

 //this polynom is used as generator of the field
 private static final int GENERATORPOLYNOMIAL=0x11B;

 /*
  * generation of look up tables can be found on
      http://www.devkb.org/java/50-AES-256-bits-encrypter-decrypter-Java-source-code
  */

 static {

     int i, j;

     Exp[0] = 1;

     for (i = 1; i < 256; i++) {

         j = (Exp[i-1] << 1) ^ Exp[i-1];

         if ((j & 0x100) != 0)

         j ^= GENERATORPOLYNOMIAL;

         Exp[i] = j;
     }

     for (i = 1; i < 255; i++)

    Log[Exp[i]] = i;
 }
```

In the above code the static block is used to generate all elements of Finite field 256. Elements of this finite field are from 0 to 255. In order to generate

all the elements of this field an irreducible polynomial is needed. Addition and subtraction in FF(256) are simply XOR operation. The methods FFMulFast and FFInv have been implemented according to the mathematical explanation of multiplication and inversve multiplication , which is given in details in chapter 3 (see section 3.3). Below you can find the code for FFMulFast method.

```
 public int FFMulFast(int firstElement, int secondElement){

 int temp=0;;

 if (firstElement==0 || secondElement==0)

 return 0;

 temp= (Log[(firstElement&0xff)] & 0xff) + (Log[(secondElement&0xff)] &
     0xff);

 if (temp>255)

 temp= temp-255;

 return Exp[(temp&0xff)];

}
```

Horner evaluation

Even if the methods "FFMulFast", "FFInv" and "add" perform mathematical operations with a desired speed performance, I decide to further improve the speed. In a scenario with files of a size of Mb or Gb calculation time is big enough. So, by using Horner's rule the number of multiplication operations is reduced significantly. Here is not described how the multiplication operations are reduced using Horner's rule. If interested in knowing more visit the site `https://www.math10.com/en/algebra/horner.html`.

The way how the Horner's rule is implemented is shown in the code below:

```
public int hornerEvaluation(int y, int []coefficientsOfPolynomial) {

 int degreeOfPolynomial=coefficientsOfPolynomial.length-1;

int result=coefficientsOfPolynomial[degreeOfPolynomial];

for (int i=degreeOfPolynomial-1; i>=0; i--) {

 result = add(coefficientsOfPolynomial[i],FFMulFast(result&0xff,y));

   }
```

```
return result;

}
```

Generation of random numbers

In full code there is a single package which handles the generation of random numbers. Each class in that package uses different algorithms for generation of random numbers. In this package *Strategy Pattern* is used. The reason is because there are different algorithms which can be used to generate random numbers. Some algorithms are better in speed performance, the others are better in security. Depending on what the clients are more concerned, is easy to choose between these algorithms.

Here the full code of this package is not shown since there are hundred lines of codes but is shown only some lines of a class which can be different in different classes. The part of the code which is not presented here for the class "RandomGeneratorSHA1" is the same to all the other classes. What is important to note is that all the classes are based on *SecureRandom class*. It is a predefined class with predefined methods and is implemented by Oracle Inc.

```
package generationOfRandomNumbers;

import java.security.NoSuchAlgorithmException;
import java.security.SecureRandom;

public class RandomGeneratorSHA1 implements randomCoeff {

    private static final int NUMBER_OF_RANDOM_BYTES = 1 ;

    private final SecureRandom csprng;

    private int holdAllRndCoefficients [] ;

    private byte[] bytes;

    public RandomGeneratorSHA1() {

      try {

         csprng = SecureRandom.getInstance("SHA1PRNG");


          } catch (NoSuchAlgorithmException e) {
          throw new RuntimeException(e);
           }

         csprng.nextBoolean();
```

```
}
```

In the code above the thing to be underlined is the line *csprng = SecureRandom.getInstance("SHA1PRNG");*

The algorithm used here is "SHA1PRNG". Use it, if speed performance is the main concern. It provides a performance speed 3 times faster than the algorithm "NativePRNG". In this package, all the classes have almost the same code with the exception of the above line. For instance, in that package there is another class named "RandomGeneratorNative". In that class, the argument of the static method "getInstance", is "NativePRNG". Use it when security is the primarily concern.

Finding the appropriate algorithm for the generation of random numbers is a very complex and a lot of work to be done. For generation of random numbers, if not an appropriate algorithm is used, it can lead to a slow speed performance and more important the generated numbers are not random at all.

Reconstruction

```
protected int LagranP(int[] coefficientValue, int[] valueOfPolynomial)
    {

        //the length of m is equal to 3 in our scenario (6,3)
       int m = coefficientValue.length;

      assert m == valueOfPolynomial.length;

      int result = 0;

      for (int i = 0; i < m; i++)
         result = ff256.add(result, ff256.FFMulFast(production(i,
             coefficientValue), valueOfPolynomial[i]));

      return result;

      }
```

In "Reconstruction" class there are three methods:

1. returnfile()

   Is a method which gives back the desired file

2. LagranP() and production() are methods which are used to evaluate *Lagrange's interpolaion formula*. From the piece of code above the *production()* method in *LagranP()* method is called. Also in this method, the methods of the class "FastFiniteField256" are being called.

5.2 Client-Server implementation

For communication between client and server a rest web service is used. In the service part PIR algorithm package is imported. It is used to do vector-matrix multiplication. In the client part shamir secret share algorithm package is imported. It is used to calculate the shares of the query vector, and to recover the desried file.

Server is used to take the shares of the files either from S3 bucket or from FileZila, the shares of the query vector, to calculate the vector-matrix multiplication, and send this result back to client. Then, client takes the result, uses returnfile method from "Reconstruction" class and sends it to the local computer from where the application is being accessed.

Client part: The client class is called "PIRClient". There are three methods in this class:

1. returnFile(). This method is a rest method. Its return type is a *Response* and its parameters are *QueryParam*. That means while typing the url of the application in a web browser the values after "?", are arguments of this method. This method calls *Reconstruction* and *Share* classes from the package of shamir algorithm. Moreover, HTTP methods are used here.

2. getIPAddress(). As its name applies, it is used to get the IP of EC2 instances.

3. getsharesFromEachistance(). As its name applies, it is used to get the shares from EC2 instances.

Server part: The sever class is called *PIRServer*. This class has three methods:

1. protocolShares(). This method is used to get the shares of the files and the shares of the vector array. Then, it uses the method *pirprotocolshares1* from *PrivateInformationRetrieval* class, to calculate vector-matrix multiplication.

2. getSharesFromFile(). As its name applies, this method is used to get the shares of the files from FileZila or S3 bucket.

3. getSecretShare(). This method is used to convert post data to integer array.

Note: The Java code of Server and Client class can be found in Appendix

PrivateInformationRetrieval class is used to implement PIR algorithm. This class has four methods.

1. allSharesForFiles(). This method is used to calculate and put in an ArrayList all shares of the files. Each element of this ArrayList is a 2D-array. ArrayList is used since it is the most preferable List when the main concern is fast iteration. Moreover, its method "add" has a complexity O(1). Complexity O(1) means that the performance will be the same as iterating only once or a thousand times.

2. allSharesForEachFile(). This method is used to take the first rows of each of the matrices above and put it into a new matrix, then take the second rows of each of the matrices above and put it into a new matrix and so on. ArrayList is also used here since its method *get* has complexity O(1)

3. pirprotocolshares1(). This method is used to calculate vector-matrix multiplication. Its return type is an array.

4. readyForReconstruction(). This method has the same result as the method above but here a 2D-array is returned. The first element for each 2D-array is used as a "key" element for returnFile() method in *Reconstruction* class.

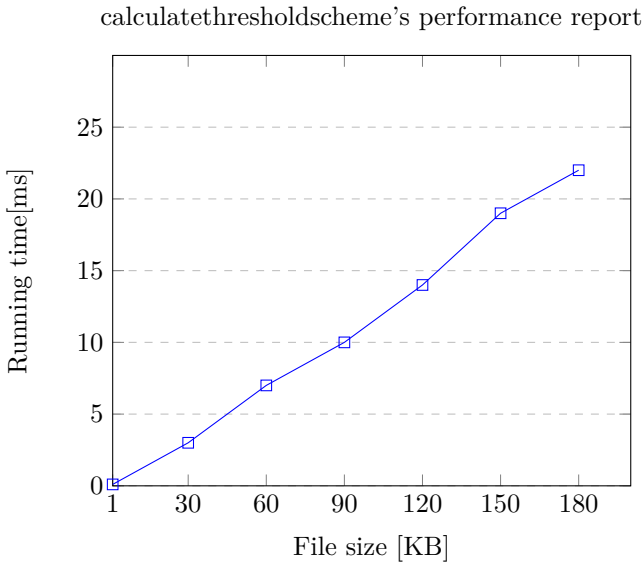
5.3 Measurements performance

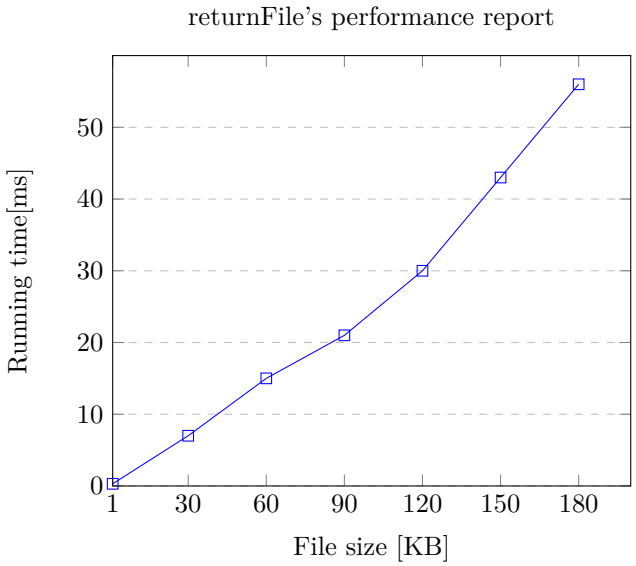
In this section performance reports of some of the methods of the application are shown. The methods to consider and which have effect in performance speed are:

1. calculatethresholdscheme (). This is used in "Share" class to calculate the shares of the file. In this method is used Horner's rule to calculate the polynomials.

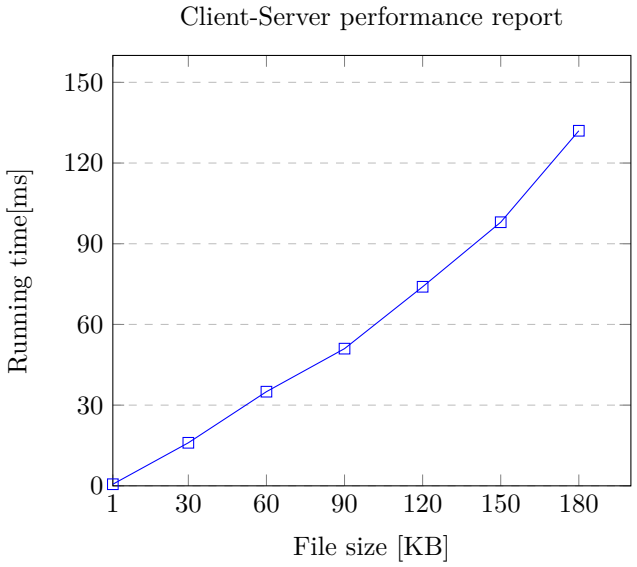
2. returnfile (). This is used in *Reconstruction* class to recover the secret.

These methods call the methods of *FastFiniteField256* class for performing all basic mathematical operations.





The first two graphs show the performace report for the *one-time activity*. The graph below shows the performance report of the real world application.



5.4 Summary

In this chapter some implementation details about the application are given. This chapter begins with implementation details about SSS algorithm. For this algorithm is necessary to implement finite field 256, to find a way how to generate random numbers, to calculate polynomials and in the end to recover the secret. For the calculation of the polynomials is used Horner's rule and is shown the way how it is implemented. Based on Lagrange interpolation, the secret is recovered. Lagrange's interpolaton implementation is shown in this chapter.

Later we talked about the Client-Server communication. We explained what is the role of each method in the client and server part. Moreover, we explained the way we have implemented PIR algorithm.

In the end, performance report for shamir algorithm are published. The average time to calculate the shares is $8Mb/s$ and the average time to recover the secret is about $3.6Mb/s$. This chapter ends with performance report of client-server communication. The average time of this communication is $1.2Mb/s$.

Chapter 6

Conclusion

In the introduction the challenges with data outsourcing to the cloud are outlined. Privacy concerns, lack of control, quality of service and availability pose risks to individual users and enterprises when adopting cloud services. An overview of Amazon Web Service cloud platform security is given. In chapter 3, are discussed the existing PIR and secret share algorithms. My application is a PIR protocol based on SSS algorithm. This algorithm provides a desired privacy level with a complexity O(n) times the size of the block, where n is the number of servers.

I am the first which have deployed that algorithm in an existing cloud platform, such as AWS. The protocol that is implemented will have a substantial; impact and broad distribution because it is flexible, compatible with existing Shamir based storage solutions,and compatible with proactive security steps in the storage systems. Programming language used in the thesis is JAVA. The main reason to implement the protocol in Java is, because in the future work I will try to integrate this application in an existing cloud platform as "Archistar".

In the end some performance measurements of the application are shown. This application can share a file with $8Mb/s$ and can reconstrut it with a speed of $3.6Mb/s$. Moreover, in a future work I will try to expand the application in a symmetric PIR.

Appendices

Appendix A

Client part

A.1 PIRClient.java

```
package com.pir.client;
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Arrays;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import PIR.PrivateInformationRetrevial;
import galoisField.FastFiniteField256;
import generationOfRandomNumbers.RandomGeneratorSHA1;
import generationOfRandomNumbers.RndGeneratorAllAlgorithm;
import secretSharing.FileToBeTaken;
```

```
import secretSharing.Reconstruction;
import secretSharing.Share;

@Path("/audit")
public class PIRClient {

   static byte[] output = null;

   static String[] ipAddress = getIPAddress();

   /*
    * service method to return file by getting shares from different
        servers
    */

   @GET
   @Path("/getFile")
   @Produces(MediaType.TEXT_PLAIN)
   public Response returnFile(@QueryParam("severs") int noOfServers,
        @QueryParam("totalfiles") int noOfFiles,
        @QueryParam("returnFileNumber") int FileNumber) throws
            IOException {

      if (noOfServers <= 0 || noOfFiles <= 0 || FileNumber <= 0
            || FileNumber > noOfFiles) {
         return Response.serverError().build();
      }

      PrivateInformationRetrevial pir = new
          PrivateInformationRetrevial();

      Reconstruction OneFileBack = new Reconstruction();

      FileToBeTaken back = new FileToBeTaken();

      Share share = new Share();

      byte[] arrayOfQueries = new byte[noOfFiles];

      arrayOfQueries[FileNumber - 1] = 1;

      FastFiniteField256 GF256 = new FastFiniteField256();

      URL url;

      DataOutputStream wr;

      HttpURLConnection conn;

      int[][] sharesOfQueries1 = share.calculateThresholdScheme(
```

```
      arrayOfQueries, noOfServers, 3, new RndGeneratorAllAlgorithm(
           new RandomGeneratorSHA1()), GF256);

String sharesOfQueries1AsString =
    Arrays.deepToString(sharesOfQueries1);

int[][] arrayForReconstructions = new int[noOfServers][];

String shareInString = null;

/*
 * This loop will connect to each EC2 service and get the shares
 */

for (int i = 0; i < noOfServers; i++) {
   url = new URL("http://" + ipAddress[i]
         + ":8080/Audit/rest/audit/protocolShares?numberOfFiles="
         + noOfFiles + "&serverNumber=" + i + "&returnFileNumber="
         + FileNumber);

   conn = (HttpURLConnection) url.openConnection();

   synchronized (conn) {
      conn.setDoOutput(true);
      conn.setRequestMethod("POST");
      conn.setRequestProperty("Accept", "text/plain");

      wr = new DataOutputStream(conn.getOutputStream());
      wr.write(sharesOfQueries1AsString.getBytes());

      wr.flush();

      wr.close();

      if (conn.getResponseCode() != 200) {
         throw new RuntimeException("Failed : HTTP error code : "
               + conn.getResponseCode());
      }

      BufferedReader br = new BufferedReader(new InputStreamReader(
           conn.getInputStream()));

      while (br.ready()) {

         shareInString = br.readLine();

      }

      int[] arrayReconstruct =
          getsharesFromEachistance(shareInString);
```

```
            arrayForReconstructions[i] = arrayReconstruct;
        }

    }


    int[][] alles =
        pir.readyForReconstruction(arrayForReconstructions);

    int[][] sharesToViewSecret11 = new int[][] { alles[0], alles[3],
          alles[1], alles[2], alles[5], alles[4] };


    output = OneFileBack.recoverfile(sharesToViewSecret11);

    // return file as attachment

    return Response
          .ok(output, MediaType.APPLICATION_OCTET_STREAM)
          .header("content-disposition",
                "attachment; filename = newfile.txt").build();

}

private static String[] getIPAddress() {

     String[] ipAddress = new String[6];

    try{

       FileReader filereader = new
           FileReader("/home/ec2-user/Audit/IpAddress.txt");

         BufferedReader bufferedreader = new BufferedReader(filereader);

         String line = bufferedreader.readLine();

         int i=0;

         while (line != null && i<6) {

                ipAddress[i]= line;

                i++;

                line = bufferedreader.readLine();

            }
```

```
      }catch(IOException e){

    }

    return ipAddress;
  }

  static int[] getsharesFromEachistance(String shareInString) {

    int[] share1 = null;

    String[] shareValues = shareInString.split(",");

    share1 = new int[shareValues.length];

    int i = 0, j = 0;

    for (String is : shareValues) {

      String removewhiteSpace = is.replaceAll("\\s", "");

      if (is.contains("]")) {

        share1[j] =
            Integer.parseInt(removewhiteSpace.replaceAll("\\]",
             ""));
        j++;

      } else if (is.contains("[")) {

        String removewhiteSpace1 = removewhiteSpace.replaceAll("\\[",
             "");

        share1[j] = Integer.parseInt(removewhiteSpace1.replaceAll(
             "\\[", ""));
        j++;

      } else {

        share1[j] =
            Integer.parseInt(removewhiteSpace.replaceAll("\\]",
             ""));
        j++;
      }
    }

    return share1;
  }
}
```

Appendix B

Server part

B.1 PIRService.java

```
package com.audit.service;

import java.io.BufferedReader;
import java.io.ByteArrayInputStream;
import java.io.DataOutputStream;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.Arrays;

import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.QueryParam;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.S3Object;
```

```
import SomeProvesForOurFullPIR.PrivateInformationRetrevial;
import galoisField.FastFiniteField256;
import generationOfRandomNumbers.RandomGeneratorSHA1;
import generationOfRandomNumbers.RndGeneratorAllAlgorithm;
import secretSharing.FileToBeTaken;
import secretSharing.Reconstruction;
import secretSharing.Share;


@Path("/audit")
public class PIRService {

   byte[] output = null;

   @POST
   @Path("/protocolShares")
   @Produces(MediaType.TEXT_PLAIN)
   public String protocolShares(@QueryParam("numberOfFiles") int files,
         @QueryParam("serverNumber") int serverNumber,
         String sharesOfQueries,
         @QueryParam("returnFileNumber") int fileNumber) {

      int[][] sharesOfQueries1 = getSecrerShare(sharesOfQueries, files,
          6);

      int[][] share1 = getSharesFromFile(files);

      int[] arrayForReconstruction = new int[share1.length];

      PrivateInformationRetrevial pir = new
          PrivateInformationRetrevial();

      arrayForReconstruction = pir.pirprotocolshares1(share1,
           sharesOfQueries1[serverNumber], fileNumber-1);

      return Arrays.toString(arrayForReconstruction);
   }

   /**
    * get the data from file and converted in to array
    *
    */

   int[][] getSharesFromFile(int numberOfFiles) {

      String shareInString = null;

      int[][] share1 = null;
```

```
try {

   /*
    * System.out.println("getting share from s3 location");
    * BasicAWSCredentials awsCreds = new
    * BasicAWSCredentials("AKIAJR2FJDU2UYMZQD4A",
    * "DFR0leDYgeBS6Vf0O04oyUHwF6qTRh9d49Pwt+dU"); AmazonS3
        s3client =
    * new AmazonS3Client(awsCreds); S3Object s3object =
    * s3client.getObject(new GetObjectRequest( bucketName,
    * "shares.txt"));
    */

   // BufferedReader br = new BufferedReader(new
   // InputStreamReader(s3object.getObjectContent()));

   BufferedReader br = null;

   br = new BufferedReader(new FileReader(new File(
         "/home/ec2-user/Audit/share/shares.txt")));

   String line;

   while ((line = br.readLine()) != null) {

      shareInString = line;

   }

   String[] shareValues = shareInString.split(",");

   for (String is : shareValues) {
   }

   share1 = new int[numberOfFiles][];

   int lenghtEachArray = 0;

   int m = 0;

   for (String is : shareValues) {

      if (is.contains("]") && m < numberOfFiles) {
         share1[m] = new int[lenghtEachArray + 1];
         m++;

         lenghtEachArray = 0;

      } else {
```

```
            lenghtEachArray++;

         }
      }

      int i = 0, j = 0;

      for (String is : shareValues) {

         String removewhiteSpace = is.replaceAll("\\s", "");

         if (is.contains("]")) {
            share1[j][i] = Integer.parseInt(removewhiteSpace
                  .replaceAll("\\]", ""));

            i = 0;

            j++;
         } else if (is.contains("[")) {

            String removewhiteSpace1 = removewhiteSpace.replaceAll(
                  "\\[", "");

            share1[j][i] = Integer.parseInt(removewhiteSpace1
                  .replaceAll("\\[", ""));
            i++;

         } else {

            share1[j][i] = Integer.parseInt(removewhiteSpace
                  .replaceAll("\\]", ""));
            i++;
         }
      }

   } catch (IOException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
   }

   return share1;
}

/**
 * converted post data to int array
 *
 */

int[][] getSecrerShare(String sharesOfQueries, int files, int
    noOfservers) {
```

```
    int[][] share1 = null;

    String[] shareValues = sharesOfQueries.split(",");

    share1 = new int[noOfservers][files + 1];
    int i = 0, j = 0;

    for (String is : shareValues) {
       String removewhiteSpace = is.replaceAll("\\s", "");

       if (is.contains("]")) {

          share1[j][i] = Integer.parseInt(removewhiteSpace.replaceAll(
                "\\]", ""));
          i = 0;

          j++;

       } else if (is.contains("[")) {

          String removewhiteSpace1 = removewhiteSpace.replaceAll("\\[",
                "");

          share1[j][i] = Integer.parseInt(removewhiteSpace1.replaceAll(
                "\\[", ""));
          i++;

       } else {

          share1[j][i] = Integer.parseInt(removewhiteSpace.replaceAll(
                "\\]", ""));
          i++;
       }
    }

    return share1;

  }

}
```

Bibliography

[1] Amazon Web Services: Overview of Security Processes *August 2015*

[2] AWS General Reference

[3] Basic Idea of PIR

[4] Grance, Mell, P. T (2009). Definition of Cloud Computing by NIST. Retrieved from NIST

[5] Native to Web API authentication privacy technique in Microsoft Azure

[6] Markus Klems, Stefan Tai, Jens Nimis, Alexander Lenk, and Thomas Sandholm. An architectural map of the cloud landscape. In Proceedings of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, *pages 23–31. IEEE Computer Society, 2009.*

[7] Cloud services models

[8] Microsft Azure Security Concepts

[9] M. Baldwin, Authentication Scenarios for Azure AD, *Updated: 02/09/2016*

[10] Cloud architecture

[11] Cloud services and deployment models, IEEE eLearning Library, Cloud Strategy Partner, LLC

[12] Overall Authentication in AWS Platform

[13] Cloud deployment models, VISMA technology

[14] Amazon Web service architecture. How it works, how can we sign up and use Amazon resourcase for creating and managing applications

[15] Cloud Academy, Advantages and Disadvantes of the cloud, Best Practices how to reduse the risk of using cloud

[16] Privacy and Data Protection by Design-from policy to engineering, European Union Agency for Network and Information Security(ENISA), *December 2014*

[17] Ian Goldberg. Improving the robustness of Private Information Retrieval. IEEE security and privacy, *pg: 131-148, 2007*

[18] Shamir Secret Sharing- Wikipedia

[19] Benny Chor, Oded Goldreich, Eyal Kushilevitz and Madhu Sudan. Private Information Retrieval. *pg : 41-50, 1995*

[20] T. Nishizeki, M. Ito and A. Saito. Secret sharing schemes realizing general access structure. Journal: Multiple assignment scheme for sharing secret. Journal of Cryptology, *6(1):15-20, 1993*

[21] Sh.Wang, X.Ding, Robert H.Deng and F.Bao. Private information retrieval using trusted hardware

[22] Finite Fields Wikipedia

[23] S.Smith and D.Safford. Practical server privacy with secure coprocessors. IBM System journal *pg:683-695, 2001*

[24] Yahoo confirms major breach — and it could be the largest hack of all time

[25] George Danezis, Josep Domingo-Ferrer, Marit Hansen, Jaap-Henk Hoepman, Daniel Le Métayer, Rodica Tirtea, Stefan Schiffner,"Privacy and Data Protection by Design – from policy to engineering".

[26] Ronald Hes and John J. Borking. Privacy-enhancing technologies: The path to anonymity. Technical report, Registratiekamer, 1995.

[27] John J. Borking. Der Identity Protector. Datenschutz und Datensicherheit, *20(11):654–658, 1996.*

[28] Casey Devet and Ian Goldberg "The Best of Both Worlds: Combining Information-Theoretic and Computational PIR for Communication Efficiency"

[29] C.Devet, I.Goldberg, N.Heninger, "Optimally Robust Private Information Retrieval"..

[30] Ian Goldberg, David R. Cheriton School of Computer Science "Improving the Robustness of Private Information Retrieval".

[31] Cloud Environment

[32] Cloud Deployment Models

[33]

[34] Jens Nimis, Markus Klems, Alexander Lenk, Thomas Sandholm, and Stefan Tai. What's inside the cloud? an architectural map of the cloud landscape. In Proceedings of the 2009 *ICSE Workshop on Software Engineering Challenges of Cloud Computing, pages 23–31. IEEE Computer Society, 2009.*

[35] Cloud Service Models

[36] Armando Fox, Gunho Lee, Rean Griffith, Ariel Rabkin, Anthony D Joseph, Randy Katz, Andy Konwinski, Michael Armbrust, David Patterson, Ion Stoica, et al. A view of cloud computing. Communications of the ACM, *53(4):50–58, 2010.*

[37] Timothy Grance and Peter Mell. The nist definition of cloud computing. NIST special publication,*800(145):7, 201*

[38] Cloud Computing downsides, Level Cloud anytime, anywhere

[39] Amazon Web Services: Overview of Security Processes *August 2015*

[40] AWS General Reference

[41] Finite Fields Wikipedia

[42] Neal R.Wagner , The laws of cryptography*2003*

[43] Microsft Azure Security Concepts

[44] M. Baldwin, Authentication Scenarios for Azure AD, *Updated: 02/09/2016*

[45] Polynomial evaluation based on Hornor's rule

[46] Rafail Ostrovski and Eyal Kushilevitz. Single Database, Computationally-Private Information Retrieval. In FOCS, *pg: 364-373, 1997*

[47] N.P.Smart and F.Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. Cryptology ePrint Archive, *2009*

[48] Adi Shamir. How to share a Secret. R.Rivest Editor, Massachusetts Institute of Technology

[49] A.Iliev and S.Smith. Private information storage with logarithm-space hardware. Internation Information Security Workshops, *pg: 199-214, 2005*

[50] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In Proceedings of the twenty-first annual ACM symposium on Theory of computing, *pages 73–85. ACM, 1989.*

[51] M.van Dijk, Sh.Halevi, C.Gentry, and V.Vaikuntanathan. Fully homomorphic encryption over the integers. *November, 2009*

[52] H. Ishizu and T. Ogihara, "A study on long-term storage of electronic data," IEICE General Conference, *2004*

[53] A.Iliev and S.Smith. Protecting client privacy with trusted computing at the server. *IEEE Security and Privacy, pg:20-28, 2005*

[54] Neal R.Wagner , The laws of cryptography*2003*

[55] Shafi Goldwasser, Benny Chor, Baruch Awerbuch and Silvio Micali. Verifiable secret sharing and achieving simultaneity in the presence of faults. In Foundations of Computer Science, 1985., *pages 383–395. IEEE, 1985.*

[56] Craig Gentry, Phd thesis, Stanford, *2009*

[57] Tobias Pulls and Daniel Slamanig. On the feasibility of (practical) commercial anonymous cloud storage. Transactions on Data Privacy, *8(2):89 111, 2015.*

[58] Amos Beimel, Secret-Sharing Schemes: A Survey, Department of Computer Science, Ben-Gurion University, Beer-Sheva, Israel.

[59] G. R. Blakley. Safeguarding cryptographic keys. In M. Smith, J. T. Zanca and R. E. Merwin editors, *volume 48, AFIPS Press, pages 313–317, 1979,.*

[60] Aguilar-Melchor, Gaborit. A Lattice-Based Computationally-Efficient Private Information Retrieval Protocol. WEWORC, *2007*

[61] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In Foundations of Computer Science, 1987., 28th Annual Symposium on, *pages 427–438. IEEE, 1987.*

[62] J. Leichter and J. Benaloh. Generalized secret sharing and monotone functions. *volume 403 of Lecture Notes in Computer Science, pages 27–35. Springer-Verlag, 1990.*

[63] E. F. Brickell. Some ideal secret sharing schemes. Journal of Combin. Math. and Combin. Comput., *6:105–113, 1989.*

[64] Y. Fujii, N. Hosaka, M.Tata, T. Kato and K. Tochikubo. "A fast (2, n)-threshold scheme and its application," *CSS2005, page 631-636, 2005*

[65] Shafi Goldwasser, Michael Ben-Or, and Avi Wigderson. Completeness theorems for noncryptographic fault-tolerant distributed computation, *pages 1–10. ACM, 1988.*

[66] K. Kurosawa and W. Ogata. "Some basic properties of general nonperfect secret sharing schemes," J. Universal Computer Science. *vol.4, no.8, page 690–704, 1998*

[67] Femi Olumofin and Ian Goldberg. Revisiting the Computational Practicality of Private Information Retrieval. *2011*

[68] Torben Pryds Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Advances in Cryptology CRYPTO 91, *pages 129–140. Springer, 1992.*