****

**Data Security in Cloud Computing**

**A PROJECT REPORT**

***Submitted by***

**SMILU THOMAS (070606126046 )**

**VINUSHAA. D (070606126055)**

**MUTHUSAMY.T (080906126062 )**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

***in***

**INFORMATION TECHNOLOGY**

**SRI SHAKTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY, COIMBATORE**

**ANNA UNIVERSITY OF TECHNOLOGY, COIMBATORE 641047**

**MAY 2011**

**ANNA UNIVERSITY OF TECHNOLOGY COIMBATORE 641047**

**BONAFIDE CERTIFICATE**

Certified that this project report “**DATA SECURITY IN CLOUD COMPUTING”** is the bonafide work of “**SMILU THOMAS (070606126046), VINUSHAA.D(070606126055)** and **MUTHUSAMY.T(080906126062)”** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Asst.prof.S.K Mouleeswaran Prof.V.Seethalakshmi

**SUPERVISOR HEAD OF THE DEPARTMENT**

Information Technology, Information Technology,

Sri Shakthi Institute of Engg & Tech., Sri Shakthi Institute of Engg & Tech.,

Coimbatore - 641062 Coimbatore - 641062

Submitted for the University Project Viva Voce conducted on \_\_\_\_\_\_\_\_\_\_\_

------------------------- -------------------------------------

**Internal Examiner External Examiner**

**ACKNOWLEDGEMENT**

We express our deep sense of gratitude to our **Chairman Dr. S. Thangavelu** for his invaluable guidance and blessings in the project work.

We are thankful to our **Secretary Mr. T. Dheepan** for his unwavering support during the entire course of this project work.

We are very grateful to our **Principal Dr. R. Manian** for providing us with an environment to complete our project successfully.

We are deeply indebted to our **HOD, Department of Information Technology, Mrs.V.Seethalakshmi** who modeled us both technically and morally for achieving greater success in life.

We express our sincere thanks to our **Project guide Mr. S.K.Mouleeswaran** for his constant encouragement and support throughout our course, especially for the useful suggestions given during the course of the project period. Also we are very grateful to our **Project Coordinator Mr. R. Ashok,** for being instrumental in the completion of our project with his complete guidance.

We are also thankful to all the staff members of our college and technicians for their valuable assistance in carrying out the project work.

Finally, we take this opportunity to extend our deep appreciation to our **family and friends,** for all that they meant to us during the crucial times of the completion of our project into a successful one.

Smilu Thomas

Vinushaa. D

Muthusamy. T

**ABSTRACT**

The Cloud Computing concept offers dynamically scalable resources provisioned as a service over the Internet. Economic benefits are the main driver for the Cloud, since it promises the reduction of capital expenditure and operational expenditure. In order for this to become reality, however, there are still some challenges to be solved. Amongst these are security and trust issues, since the user’s data has to be released to the Cloud and thus leaves the protection sphere of the data owner. In contrast to traditional solutions, where the IT services are under proper physical, logical and personnel controls, Cloud Computing moves the application software and databases to the large data centers, where the management of the data and services may not be fully trustworthy. This unique attribute, however, poses many new security challenges which have not been well understood. In this project, we focus on cloud data storage security, which has always been an important aspect of quality of service. Cloud computing delivers convenient, on-demand access to shared pools of data, applications and hardware over the internet. Cloud computing provides unlimited infrastructure to store and execute customer data and program. As customers we do not need to own the infrastructure, they are merely accessing or renting; they can forego capital expenditure and consume resources as a service, paying instead for what they use. Data can be redundantly store in multiple physical locations. Due to this redundancy the data can be easily modified by unauthorized users which can be stored in the database. This leads to loss of data privacy and security to database. Extensive security and performance analysis shows that the proposed scheme ensures that cyclic redundancy check and time-tested practices and technologies for managing trust relationships in traditional enterprise IT environments can be extended to work effectively in both private and public clouds. Those practices include data encryption, strong authentication and fraud detection.

**CHAPTER NO. TITLE PAGE NO.**

**1 Introduction**

1.1 Overview

**2 System Analysis**

2.1 Existing System

2.2 Proposed System

2.2.1 Advantages of Proposed System

2.3 Feasibility Study

2.3.1 Economical Feasibility

2.3.2 Behavioral Feasibility

2.3.3 Technical Feasibility

**3 System Specification**

3.1 Hardware Requirements

3.2 Software Requirements

3.2.1 ASP .NET

3.2.2 MSSQL

**4 Software Description**

4.1 Front End

4.2 ASP.NET 2.0 Features

4.3 Back End

4.4 MSSQL Features

**5 Project Description**

5.1 Problem Definition

5.2 Overview of the Project

5.3 Module Description

5.3.1 Module 1: Key Generation by TPA

5.3.2 Module 2: Encryption and Decryption of

data using DES algorithm

5.3.3 Module 3: Error localization

5.4 Data Flow Diagram

5.5 E-R Diagram

5.6 Database Design

5.6.1 Table 1

5.6.2 Table 2

5.6.3 Table 3

5.7 Input Design

5.8 Output Design

**6 System Testing**

6.1 Unit Testing

6.2 Integration Testing

6.3 Acceptance Testing

6.4 Test Cases

**7 System Implementation**

**8 Conclusion & Future Enhancements**

8.1 Conclusion

8.2 Future Enhancements

**9 Appendix**

9.1 Source Code

9.2 Screen Shots

**10 References**

**CHAPTER 1**

**INTRODUCTION**

* 1. **Overview**

Several trends are opening up the era of Cloud Computing, which is an Internet-based development and use of computer technology. The ever cheaper and more powerful processors, together with the software as a service computing architecture, are transforming data centers into pools of computing service on a huge scale. The increasing network bandwidth and reliable yet flexible network connections make it even possible that users can now subscribe high quality services from data and software that reside solely on remote data centers. Moving data into the cloud offers great convenience to users since they don’t have to care about the complexities of direct hardware management. From the perspective of data security, which has always been an important aspect of quality of service, Cloud Computing inevitably poses new challenging security threats for number of reasons. **Firstly**, traditional cryptographic primitives for the purpose of data security protection can not be directly adopted due to the users’ loss control of data under Cloud Computing. Therefore, verification of correct data storage in the cloud must be conducted without explicit knowledge of the whole data. **Secondly,** Cloud Computing is not just a third party data warehouse. The data stored in the cloud may be frequently updated by the users, including insertion, deletion, modification, appending, reordering, etc. To ensure storage correctness under dynamic data update is hence of paramount importance. However, this dynamic feature also makes traditional integrity insurance techniques futile and entails new solutions. **Finally**, the deployment of Cloud Computing is powered by data centers running in a simultaneous, cooperated and distributed manner. Individual user’s data is redundantly stored in multiple physical locations to further reduce the data integrity threats. Therefore, distributed protocols for storage correctness assurance will be of most importance in achieving a robust and secure cloud data storage system in the real world. However, such important area remains to be fully explored in the literature.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 Existing System**

The main problems existing in present systems are:

Byzantine Failure (i.e.) No consistency is guaranteed, different users may see different values of the output .Malicious data modification attack server colluding attacks. Current system eliminates the responsibility of local machines for data maintenance. Users are at the mercy of their cloud service providers for the availability and integrity of their data. Many of predecessors only provide binary results about the storage state across the distributed servers and not localization of data errors.

**2.2 Proposed System**

We focus on cloud data storage security, which has always been an important aspect of quality of service. To ensure the correctness of users’ data in the cloud, we propose an effective and flexible distributed scheme with two salient features, opposing to its predecessors. By utilizing the homomorphic token our scheme achieves the integration of storage correctness insurance and data error localization, i.e., the identification of misbehaving server(s). Unlike most prior works, the new scheme further supports secure and efficient dynamic operations on data blocks, including: data update, delete and append.

**2.2.1 Advantages of Proposed System**

* Storage correctness: to ensure users that their data are indeed stored appropriately and kept intact all the time in the cloud.
* Fast localization of data error: to effectively locate the malfunctioning server when data corruption has been detected.
* Dynamic data support: to maintain the same level of storage correctness assurance even if users modify, delete or append their data files in the cloud.
* Dependability: to enhance data availability against Byzantine failures, malicious data modification and server colluding attacks, i.e. minimizing the effect brought by data errors or server failures.
* Lightweight: to enable users to perform storage correctness checks with minimum overhead.

**2.3 Feasibility Study**

The development and implementation of a new system is definitely expensive. It requires system resources, manpower, time and money, so it improves the necessity of the feasibility study based on the proposed system requirements. During system analysis, the feasibility study of the proposed system was carried out. The main objective of this study is to determine whether the proposed system is feasible or not i.e. to ensure that proposed system is not a burden to the organization. And it was concluded that our project is feasible.

The study was categorized into three types. They are:

* **Economic Feasibility**
* **Behavioral Feasibility**
* **Technical Feasibility**

**2.3.1 Economic Feasibility study**

This study is carried out to check the economic impact that the system will have the organization. The technique of cost benefit analysis is often used a basis for assessing economic feasibility. The “**Data Security in Cloud Computing**” can be developed at a reasonable cost with the available hardware, software and manpower. The study include the number of systems required, Software cost and its availability and number of man hours required to complete the project. So projects benefits overweigh the cost.

**2.3.2 Behavioral Feasibility study**

This aspect of study is to check the level of acceptance of the system by the user. The levels of the acceptance by the users solely depend on the methods that are employed to educate the user about the system and to make them familiar with it. Their level of confidence must be raised so that they are also able to make some constructive criticism, which is welcomed, as they are the final user of the system. In data security in clouod computing the users can save the data and files in the cloud server provided by the Clousd Service Provider and can check for the correctness of data. Third party Auditor assists the user in transferring the data safely across the cloud server based on Service Level agreement.

**2.3.3** **Technical Feasibility study**

This study is carried out to check the technical facility, i.e. the technical requirements of the system. The assessment of technical feasibility was based on an outline design of system requirements in terms of input, output, files, programs and procedures. This was qualified in terms of volumes of data, trends, frequency of updating, cycles of activity etc, in order to give an introduction of technical system. The system should thus provide technical guarantee of accuracy, reliability, ease of access and data security and through this the hardware and software requirements should be satisfied. The developed system has a modest technical requirement, as only minimal or null changes are required in implementing the system.

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**HARDWARE REQUIREMENT**

Processor - Pentium IV

Hard Disk - 20 GB

RAM - 512 MB

Monitor - 15 Inches Color Monitor

CD/DVD Drive - LG DVD RW/W

Keyboard - Mercury Standard Keyboard

Mouse - Logitech

**SOFTWARE REQUIREMENT**

Operating System - Windows 200x/XP and above version

Application Tool - ASP.NET 2.0

Web Server - Internet Information Services 5.1

Database Tool - MS SQL Server 2000

**CHAPTER 4**

**SOFTWARE DESCRIPTION**

**4.1 FRONT END**

In client/server applications, the client part of the program is often called the front end and the server part is called the back end. The front end is responsible for collecting input in various forms from the user and processing it to conform to a specification the back end can use. The front end is an [interface](http://en.wikipedia.org/wiki/Interface_%28computer_science%29) between the user and the back end. The front end used in our project is ASP.NET 2.0 (C#) which is supported in windows platform.

**4.2 ASP.NET 2.0**

**History**

Before organizations were even thinking about developing applications for the Internet, much of the application development focused on thick desktop applications. These thick-client applications were used for everything from home computing and gaming to office productivity and more. No end was in sight for the popularity of this application model. During that time, Microsoft developed its thick-client applications using mainly Visual Basic (VB). Visual Basic was not only a programming language; it was tied to an IDE that allowed for easy thick-client application development. In the Visual Basic model, developers could drop controls onto a form, set properties for these controls, and provide code behind them to manipulate the events of the control. For example, when an end user clicked a button on one of the Visual Basic forms, the code behind the form handled the event. Then, in the mid-1990s, the Internet arrived on the scene. Microsoft was unable to move the Visual Basic model to the development of Internet-based applications. The Internet definitely had a lot of power, and right away the problems facing the thick-client application model were revealed. Internet-based applications created a single instance of the application that everyone could access. Having one instance of an application meant that when the application was upgraded or patched, the changes made to this single instance were immediately available to each and every user visiting the application through a browser.

**The Goals of ASP.NET 2.0**

ASP.NET 2.0 is a major release of the product and is an integral part of the .NET Framework 2.0. This release of the Framework was code-named Whidbey internally at Microsoft. You might hear others referring to this release of ASP.NET as ASP.NET Whidbey. ASP.NET 2.0 heralds a new wave of development that should eliminate any of the remaining barriers to adopting this new way of coding Web applications.

When the ASP.NET team started working on ASP.NET 2.0, it had specific goals to achieve. These goals focused around developer productivity, administration and management, as well as performance and scalability. These goals are achieved with this milestone product release. The next sections look at each of these goals.

**Features of ASP.NET 2.0**

**Developer Productivity**

Much of the focus of ASP.NET 2.0 is on productivity. Huge productivity gains were made with the release of ASP.NET 1.x—could it be possible to expand further on those gains.

One goal the development team had for ASP.NET 2.0 was to eliminate much of the tedious coding that ASP.NET originally required and to make common ASP.NET tasks easier. The ASP.NET team developing ASP.NET 2.0 had the goal of reducing by two-thirds the number of lines of code required for an ASP.NET application! It succeeded in this release; you will be amazed at how quickly you can create your applications in ASP.NET 2.0.

**Administration and Management**

The initial release of ASP.NET focused on the developer, and little thought was given to the people who had to administer and manage all the ASP.NET applications that were built and deployed. Instead of working with consoles and wizards as they did in the past, administrators and managers of these new applications now had to work with unfamiliar XML configuration files such as machine.config and web.config.

To remedy this situation, ASP.NET 2.0 now includes a Microsoft Management Console (MMC) snap-in that enables Web application administrators to edit Configuration settings easily on the fly. Configuration Settings dialog allows system administrators to edit the contents of the machine.config and the web.config files directly from the dialog instead of having them examine the contents of an XML file.

**Performance and Scalability**

One of the goals for ASP.NET 2.0 set by the Microsoft team was to provide the world’s fastest Web application server. This book also addresses a number of performance enhancements available in ASP.NET 2.0. One of the most exciting performance enhancements is the new caching capability aimed at exploiting Microsoft’s SQL Server. ASP.NET 2.0 now includes a feature called SQL cache invalidation. Before ASP.NET 2.0, it was possible to cache the results that came from SQL Server and to update the cache based on a time interval—for example, every 15 seconds or so. This meant that the end user might see stale data if the result set changed sometime during that 15-second period.

In some cases, this time interval result set is unacceptable. In an ideal situation, the result set stored in the cache is destroyed if any underlying change occurs in the source from which the result set is retrieved—in this case, SQL Server. With ASP.NET 2.0, you can make this happen with the use of SQL cache invalidation. This means that when the result set from SQL Server changes, the output cache is triggered to change, and the end user always sees the latest result set. The data presented is never stale. another big area of change in ASP.NET is in the area of performance and scalability. ASP.NET 2.0 now provides 64-bit support. This means that you can now run your ASP.NET applications on 64-bit Intel or AMD processors. Because ASP.NET 2.0 is fully backward compatible with ASP.NET 1.0 and 1.1, you can now take any former ASP.NET application, recompile the application on the .NET Framework 2.0, and run it on a 64-bit processor.

**New Developer Infrastructures**

An exciting advancement in ASP.NET 2.0 is that new infrastructures are in place for you to use in your applications. The ASP.NET team selected some of the most common programming operations performed with ASP.NET 1.0 to be built directly into ASP.NET. This saves you considerable time and coding.

**Membership and Role Management**

In earlier versions, if you were developing a portal that required users to log in to the application to gain privileged access, invariably you had to create it yourself. It can be tricky to create applications with areas that are accessible only to select individuals.

**Personalization**

One advanced feature that portals love to offer their membership base is the capability to personalize their offerings so that end users can make the site look and function however they want. The capability to personalize an application and store the personalization settings is now completely built into the ASP.NET framework.

Because personalization usually revolves around a user and possibly a role that this user participates in, the personalization architecture can be closely tied to the membership and role infrastructures. You have a couple of options for storing the created personalization settings. The capability to store these settings in either Microsoft Access or in SQL Server is built into ASP.NET 2.0. As with the capabilities of the membership and role APIs, you can use the flexible provider model, and then either change how the built-in provider uses the available data store or build your own custom data provider to work with a completely new data store. The personalization API also supports a union of data stores, meaning that you can use more than one data store if you want. Because it is so easy to create a site for customization using these new APIs, this feature is quite a value add for any application you build.

**The ASP.NET Portal Framework**

During the days of ASP.NET 1.0, developers could go to the ASP.NET team’s site and download some Web application demos called IBuySpy., These demos were known as Developer Solution Kits and are used as the basis for many of the Web sites on the Internet today. Some were even extended into Open Source frameworks such as DotNetNuke.

The nice thing about IBuySpy was that you could use the code it provided as a basis to build either a Web store or a portal. You simply took the base code as a starting point and extended it. For example, you could change the look and feel of the presentation part of the code or introduce advanced functionality into its modular architecture. Developer Solution Kits were quite popular because they made performing these types of operations so easy.

**Site Navigation**

The ASP.NET team members realize that end users want to navigate through applications with ease. The mechanics to make this work in a logical manner is sometimes hard to code. The team solved the problem in ASP.NET 2.0 with a series of navigation-based server controls.

**4.3 BACK END**

A "back-end" application or program serves indirectly in support of the front-end services, usually by being closer to the required resource or having the capability to communicate with the required resource. The back-end application may interact directly with the front-end or, perhaps more typically, is a program called from an intermediate program that mediates front-end and back-end activities.

**4.4 MSSQL SERVER 2000**

**A**s a database, SQL Server is all about efficiently storing data within tables built from rows and columns. At the center of SQL Server is the SQL Server engine, which processes the database commands. The process runs inside windows and understands only connections and SQL commands. Enterprise Manager, Query Analyzer, every SQL Server–enabled Graphical User Interface (GUI), Application Programming Interface (API), and application makes a connection to SQL Server and sends SQL statements to SQL Server for processing. As robust as the engine is, SQL Server is much more than just the engine: it includes a set of tools for administrating the server and preparing queries; add-on tools for converting and moving data, and for performing data warehousing and analysis; and services for managing the connection at both the server side and the client side. SQL Server is based on the ANSI SQL 92 standard. SQL is the de facto standard for stating relational-database queries. Nearly every database product is based on some variation of SQL, even if the SQL code is not visible to the end user.

SQL Server is a complete database system, and fully mastering its scope can take years. In terms of features, commands, subsystems, components, and possibilities, SQL Server is one of the largest and most complex software products on the market. Fortunately, Microsoft has gone the extra mile to improve the “out-of-the-box experience” by making SQL easier to use and administer than other client/server database systems (including previous version of SQL Server). The server administration can be so simple that I know of several databases in production that were set up using administrative wizards and have not required administrative attention in over two years, yet they’re still running great. Nonetheless, the sheer number of SQL Server features can easily overwhelm a new developer.

**The Advantages of SQL Server**

SQL Server is growing in popularity for good reason—it’s a database with numerous compelling advantages. Microsoft marketing may offer a different set of reasons, but in the following sections I introduce what I think are the features that make SQL Server so great (in order of importance).

**ACID Properties and High Availability**

Data integrity is the single most important feature of any significant database. To me, the single greatest benefit of SQL Server is its rock-solid implementation of the ACID properties— which means that transactions are Atomic, Consistent, Isolated, and Durable. Heavy-duty databases are judged by their implementation of the ACID properties. Closely related to ACID, the concept of *availability* in this context means that the data remains available even in the face of trouble. SQL Server uses a write-ahead transaction log, robust recovery methods, and high-end features such as log shipping and clustered servers to provide high availability. It will cost a pretty penny, but if it’s important that the database always be available, regardless of the situation, SQL Server can do it. Even with standard server hardware, SQL Server data is highly available and easily recoverable.

**SQL Server Has Become the Standard**

If there’s one thing I’ve learned in two decades of developing databases, it’s that staying current in the software industry is like river rafting. Let me explain. The water in a river is constantly moving, but if you don’t stay in the center of the current, you can get stuck in an eddy or hung up in the side of a rapid. Technology has to be more than cool to be desirable (remember NeXT computers?); it has to have momentum that will carry it through to tomorrow. The choice is to be in the current or to be caught in the debris.

SQL Server qualifies as a fast-moving current in five significant ways:

* SQL Server has the sales numbers to demonstrate it’s a standard that’s going to be here for a while. SQL Server sales surpassed the billion-dollar mark in mid-2001, and SQL Server is outselling the competition in both dollar volume and units of sales. SQL Server is the most popular Windows client/server database with 38 percent to 70 percent of market share (depending on whose figures are most believable), and Windows databases are growing twice as fast as UNIX databases. SQL Server holds a 68 percent market share of Web databases.
* Bill Gates recently identified SQL Server as one of Microsoft’s most important products.
* SQL Server is a true relational database and is entry-level compliant with the SQL 92 ANSI standard.
* SQL Server is capable of supporting very large databases (VLDBs) having 1000s of users and terabytes of data. Several corporate success stories about very large databases running on SQL Server 2000

**SQL Server Security**

Critical data must be secure. If there’s a chance that any of the data has been compromised then all the data is suspect. When initially installed, SQL Server is wide open, which is OK because it makes learning SQL Server much easier. But when you want to secure the data, SQL Server has a very clean security model, which can be configured to meet the U.S. federal C2 security requirements.

**SQL Server Performance and Scalability**

Many developers are being introduced to SQL Server from Microsoft Access with the goal of improving performance for an existing database application. If that’s you, you’ll be pleased with SQL Server. It’s fast and highly tunable. From my experience, a nice dual CPU server with a couple of RAID 5 disk subsystems and half a gig of RAM is incredibly fast.

While the performance of desktop databases will eventually max out, SQL Server performance will continue to scale with additional hardware. Scalability is the sustained performance as the database grows from a small database (under 10GB) to a large database (over 100GB), moving from Windows CE to multimillion-dollar 32-CPU clusters. It can take advantage of the hardware and add high-end features to handle very large databases.

The TCP-C benchmark is designed to simulate a typical online transaction processing (OLTP) database. The benchmark specifications detail the database schema and transactions for a standard order processing/inventory system, the way in which the costs must be calculated, and the auditing procedures for measuring the performance of the database. The TPC-C throughput measures how many new orders the database can accept while simultaneously handling four other types of transaction (payment, order-status, delivery, and stock-level).

**Balanced and Complete**

Several types of database applications exist. Transactional applications that handle the day-to-day work of a business are known as On-Line Transactional Processing Applications (OLTPs) and require the transactional-integrity aspects of the ACID properties mentioned earlier. Performance is critical for a database handling thousands of updates. SQL Server is well tuned for these applications, as are several other database products.

Another type of database application is intended to gather huge amounts of data history and perform amazing feats of analysis, data mining, and trend identification. These are On-Line Analysis Processing (OLAP) database applications. While several third-party analysis products are available, SQL Server includes tools for gathering data into SQL Server and performing the analysis.

The balanced and complete capabilities of SQL Server are one of its advantages. The numerous additional features, such as replication, DTS, Analysis Services, and jobs provide a future for a SQL Server–based project.

**Developer Flexibility**

SQL Server developers and DBAs enjoy a wide variety of interfaces and levels. For many databases, SQL Server’s automated default settings will work fine, but the control will be there when needed. As a developer or DBA, you decide the amount of control appropriate for the project. If you don’t want to be bothered, SQL Server can handle most of the administration automatically. SQL Server offers multiple interfaces. If you prefer down-and-dirty code, you can control almost every feature of SQL Server without ever seeing a graphical interface. At the other end of the spectrum, even complex tasks can be accomplished with one of the 22 major wizards. SQL Server offers you several ways to accomplish any task. Even within a SQL query there’s the flexibility to state a request in the way that makes the most sense.

**Price and Performance**

SQL Server is the cheapest of the high-end client/server databases based on the TCP benchmark cost per transaction (www.tcp.org). SQL Server 2000’s cost per transaction is 44 percent less than the average cost per transaction of the other configurations in the TCP benchmark top-ten list. Microsoft has priced SQL Server with a variety of editions and licensing models.

**CHAPTER 5**

**PROJECT DESCRIPTION**

**5.1 Problem Definition**

Security threats faced by cloud data storage can come from two different sources.

On the one hand, a CSP can be self-interested, untrusted and possibly malicious. Not only does it desire to move data that has not been or is rarely accessed to a lower tier of storage than agreed for monetary reasons, but it may also attempt to hide a data loss incident due to management errors.

On the other hand, there may also exist an economically motivated adversary, who has the capability to compromise a number of cloud data storage servers in different time intervals and subsequently is able to modify or delete users’ data while remaining undetected by CSPs for a certain period. Specifically, we consider two types of adversary with different levels of capability:

**Weak Adversary**: The adversary is interested in corrupting the user’s data files stored on individual servers. Once a server is comprised, an adversary can pollute the original data files by modifying or introducing its own fraudulent data to prevent the original data from being retrieved by the user.

**Strong Adversary**: This is the worst case scenario, in which we assume that the adversary can compromise all the storage servers so that he can intentionally modify the data files as long as they are internally consistent. In fact, this is equivalent to the case where all servers are colluding together to hide a data loss or corruption incident.

**5.2 Overview of the Project**



Representative network architecture for cloud data storage is illustrated in Figure. Three different network entities can be identified as follows:

*•***User**: users, who have data to be stored in the cloud and rely on the cloud for data computation, consist of both individual consumers and organizations.

*•* **Cloud Service Provider (CSP):** a CSP, who has significant resources and expertise in building and managing distributed cloud storage servers, owns and operates live Cloud Computing systems.

*•* **Third Party Auditor (TPA):** an optional TPA, who has expertise and capabilities that users may not have, is trusted to assess and expose risk of cloud storage services on behalf of the users upon request.

**5.3 Module Description**

**5.3.1 Module 1: Key Generation by TPA**

**Key generation** is the process of generating keys for cryptography. A key is used to encrypt and decrypt. TPA is third party auditor who generates the key as per the user requests.

There are three keys used for key generation.

**Random key:**

It’s a key generated using next function. It generates a random key specified within the range .This key is used for session key generation .Each time the request is send a new random key will be generated and hence it is known as pseudo random key.

**Session key:**

A symmetric procedure avoids the problems with prior art systems using modifiers with master keys and generates a secure session key from a secret master key and an additional pair of randomly selected signals. The secret master key is known to both parties: one at station A and one at station B. One randomly selected signal of the pair is generated by the party at station A while the other signal in the pair is generated by the party at station B. In one embodiment, a random number signal sent by each one of the parties to the other is encrypted before transmission and decrypted upon reception. Both encryption (at one station) and decryption (at the other station) employ symmetric key cryptographic systems which use the secret master key. The session key is then formed by a commutative combination of both random number signals.

**Update key:**

It’s the final key generated by the TPA based on the random key ,session key and user name and password. User name and password are converted into ASCII values and then used for key generation. This key is displayed by the TPA to the user. In real world problems it will be kept hidden in the user interface

Trusted third party can be selected by the user themselves based on the following functionalities

* + TPA’s reputation and history.
  + Quality of services provided to other users
  + Quality assurance and security management standards currently followed by the company (*e.g*. certified compliance with ISO 9000 and ISO/IEC 27001).

### A formal contract between TPA and the usersr shall exist to protect both parties.The contract shall clearly define the types of information exchanged and the purpose for so doing.

### Technical access controls shall include:

* User identification and authentication;
* Authorization of access, generally through the assignment of users to defined user roles having appropriate logical access rights and controls;
* Data encryption in accordance with user’s encryption policies and standards defining algorithms, key lengths, key management and escrow *etc*.
* Accounting/audit logging of access checks, plus alarms/alerts for attempted access violations where applicable.

**5.3.2 Module 2: Encryption and Decryption of data using DES algorithm**

Data that is send to the Server by different users are encrypted and stored using Data Encryption Standard (DES) Algorithm.

Steps for Encryption:

* Must be 64 bits, 8 bytes key
* Distribute this key to the user who will decrypt this file.
* Get the Key for the file to Encrypt.
* Encrypt the file.
* Remove the Key from memory.

Steps for Decryption:

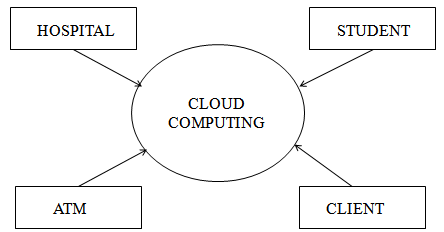
* Create a file stream to read the encrypted file back.
* Decrypt the data using the key.
* Remove the key from memory.

**5.3.3 Module 3: Error localization**

The error in the stored data is found by the TPA using miscellaneous dump code and it will say in which part the error has occurred.

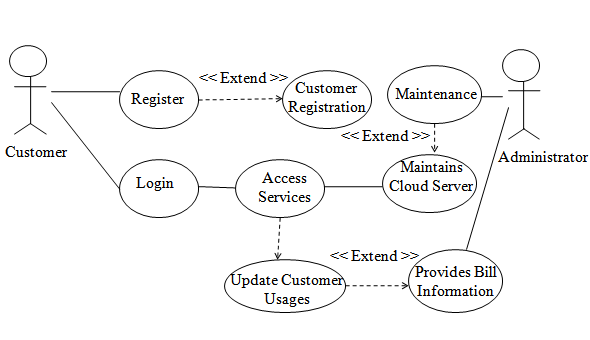
**5.4 Data Flow Diagram**

**First Level DFD**



**5.5 System Flow Chart**

**5.6 Use Case Diagram**

****

**5.6 Database Design**

**5.6.1 Table 1**

**Adminmaster**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Constraint** |
| AdminID | Int | 4 | Primary Key |
| AdminUserName | Varchar | 50 |  |
| AdminPassword | Varchar | 50 |  |

**5.6.2 Table**

**Hospital Details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Constraint** |
| HospitalID | Int | 4 | Primary Key |
| HospitalName | Varchar | 100 |  |
| Addressline1 | Varchar | 100 |  |
| Addressline2 | Varchar | 100 |  |
| City | Varchar | 100 |  |
| PinCode | Varchar | 20 |  |
| State | Varchar | 100 |  |
| CountryID | Int | 4 | Foreign Key |
| ContactNumber | Varchar | 50 |  |
| EmailAddress | Varchar | 100 |  |

**5.6.3 Table 3**

**Atmdetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Constraint** |
| AtmID | Int | 4 | Primary Key |
| AtmCenterName | Varchar | 100 |  |
| BankName | Varchar | 50 |  |
| TransactionDate | Datetime | 15 |  |
| CityName | Varchar | 100 |  |
| CountryID | Int | 4 | Foreign Key |
| CashDeposited | Numeric | 10,2 |  |
| CashWithdrawn | Numeric | 10,2 |  |

**5.6.4 Table 4**

**Studentdetails**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fieldname** | **Datatype** | **Length** | **Constraint** |
| StudentID | Int | 4 | Primary Key |
| StudentName | Varchar | 100 |  |
| CourseName | Varchar | 50 |  |
| YearName | Varchar | 20 |  |
| LevelName | Varchar | 20 |  |
| CollegeName | Varchar | 100 |  |
| City | Varchar | 100 |  |
| CountryID | Int | 4 | Foreign Key |
| ContactNumber | Varchar | 50 |  |
| EmailAddress | Varchar | 100 |  |

**5.7 Input Design**

Input facilities the entry of data into the computer system. Input design involves the selection of the best strategy for getting data into the computer system at the right time and as accurately as possible. This is because the most difficult aspect of input design in accuracy .The use of well-defined documents can encourage users to record data accurately without omission. Input design must capture all the data that the system needs, without introducing any errors. Input errors can be greatly reduced when inputting directly by using appropriate forms for data capture and well designed computer screen layout.

System analysts decide the following input design details like, what data item to input, what medium to use, how the data should be arranged or coded data items and transaction needing validations to detect errors and at last the dialogue to guide users in providing input. Input data of a system may not be necessarily a raw data captured in the system from scratch. These can also be the output of another system or sub-system. The design of input covers all phases of input from the certain of initial data to actual entering the data to the system for processing.

**Input provided for our system:**

* User name and password that generates key for authentication of users
* User data that need to transferred securely.

**5.8 Output Design**

Output design generally refers to the results and information that are generated by the system. For many end-users, output is the main reason for developing the system and the basis on which they evaluate the usefulness of the application and its result. The objective of a system finds its shape in terms of output. The analysis of the objective of a system leads to determination of outputs .Outputs of a system can take various forms. The most common are reports, screens displays printed form, graphical drawing etc. the outputs vary in terms of their contents, frequency and timing. The users of the output, its purpose and sequence of details to be printed are all considered for input.

The output from a system is the justification for its existence. If the outputs are inadequate in anyway, the system itself is inadequate. The basic requirements of output are that it should be accurate, timely and appropriate, in terms of content, medium and layout for its intended purpose. Hence it is necessary to design output so that the objectives of the system are met in the best possible manner. The outputs are in the form of web page displayed in web browsers when the URL is entered. When designing output, the system analyst must accomplish things like, to determine what information to be present, to decide whether to display or print the information and select the output medium to distribute the output to intended recipients.

**CHAPTER 6**

**SYSTEM TESTING**

**6.1** **Unit Testing**

The procedure level testing is made first. By giving improper inputs, the errors occurred are noted and eliminated. Then the web form level testing is made.

* Storage of data to the table in the correct manner. In the administrator as well as user registration form, the zero length username and password are given and checked. Also the duplicate username is given and checked. Correct username and wrong password and vice versa is also checked which gives the error message as expected.
* In registration form, the button will send data to the server, if the client side validations are made. The dates are entered in wrong manner and checked.
* Wrong email-id and web site URL (Universal Resource Locator) is given and checked.

**6.2 Integration Testing**

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Even if a software component is successfully unit tested, in an enterprise n-tier distributed application it is of little or no value if the component cannot be successfully integrated with the rest of the application. Once unit tested components are delivered we then integrate them together.   
These “integrated” components are tested to weed out errors and bugs caused due to the integration. This is a very important step in the Software Development Life Cycle.

Test data includes username and password used for user authentication. Data that are entered by the user for different services such as ATM etc…Different username and password are given as input and checked if each time different key is generated by the TPA. When not valid username and password is given the key is not generated by the TPA and an error message is shown. Checked data is encrypted and decrypted correctly.

Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

**6.3 Validation Testing**

The final step involves Validation testing, which determines whether the software function as the user expected. The end-user rather than the system developer conduct this test most software developers as a process called “Alpha and Beta Testing” to uncover that only the end user seems able to find. The end users enter all the data at the user side and check if their data are stored and retrieved correctly and if any error in their data it is informed to them. They are not concerned of the back end processes.

**CHAPTER 7**

**IMPLEMENTATION**

Implementation is the most crucial stage in achieving a successful system and giving the user’s confidence that the new system is workable and effective. Implementation is a modified application to replace an existing one. This type of conversation is relatively easy to handle, provide there are no major changes in the system.

Each program is tested individually at the time of development using the data and has verified that this program linked together in the way specified in the programs specification, the computer system and its environment is tested to the satisfaction of the user. The system that has been developed is accepted and proved to be satisfactory for the user. And so the system is going to be implemented very soon. A simple operating procedure is included so that the user can understand the different functions clearly and quickly.

Initially as a first step the executable form of the application is to be created and loaded in the common server machine which is accessible to the entire user and the server is to be connected to a network. The next step is to sign a Service Level Agreement between Service providers and TPA. It is followed by saving the data in server and it is used by the client. The final stage is to document the entire system which provides components and the operating procedures of the system.

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1 Conclusion**

Cloud computing is an emerging computing technology that uses the Internet and central remote servers to host data and applications. It allows consumers and businesses to use applications without installing them locally and access information from any computer with Internet access. Cloud computing allows for much more efficient computing by using centralized storage, memory, and processing. The benefits of cloud computing are clear, and so is the need to develop appropriate security for cloud implementations. It has many benefits, such as better hardware management, since all the computers are the same and run the same hardware. It also provides for better and easier management of data security, since all the data is located on a central server, so administrators can control who has and doesn't have access to the files.

**8.2 Future Enhancements**

We believe that data storage security in Cloud Computing, an area full of challenges and of paramount importance, is still in its infancy now, and many research problems are yet to be identified. We envision several possible directions for future research on this area. The most promising one we believe is a model in which public verifiability is enforced. Public verifiability, allows TPA to audit the cloud data storage without demanding users’ time, feasibility or resources. An interesting question in this model is if we can construct a scheme to achieve both public verifiability and storage correctness assurance of dynamic data. Besides, along with our research on dynamic cloud data storage, we also plan to investigate the problem of fine-grained data error localization. Cloud Computing is a vast topic and the above report does not give a high level introduction to it. It is certainly not possible in the limited space of a report to do justice to these technologies. What is in store for this technology in the near future? Well, Cloud Computing is leading the industry’s endeavor to bank on this revolutionary technology.

Cloud Computing Brings Possibilities

* Increases business responsiveness
* Accelerates creation of new services via rapid prototyping capabilities
* Reduces acquisition complexity via service oriented approach
* Uses IT resources efficiently via sharing and higher system utilization
* Reduces energy consumption
* Handles new and emerging workloads
* Scales to extreme workloads quickly and easily
* Simplifies IT management
* Platform for collaboration and innovation
* Cultivates skills for next generation workforce

Today, with such cloud-based interconnection seldom in evidence, cloud computing might be more accurately described as "sky computing," with many isolated clouds of services which IT customers must plug into individually. On the other hand, as virtualization and SOA permeate the enterprise, the idea of loosely coupled services running on an agile, scalable infrastructure should eventually make every enterprise a node in the cloud. It's a long-running trend with a far-out horizon. But among big megatrends, cloud computing is the hardest one to argue with in the long term.

Cloud Computing is a technology which took the software and business world by storm. The much deserved hype over it will continue for years to come.

**CHAPTER 10**

**REFERENCES**

**[1]** Ensuring Data Storage Security in cloud computing by Cong Wang, Qian Wang, Kui Ren and Wenjing Lou,IEEE Computer Society December 2010

**[2]** E. Shi, J. Bethencourt, T. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In IEEE Symposium on Security and Privacy, pages 350{364, Washington, DC, USA, 2007. IEEE Computer Society.

**[3]** Amazon.com, “Amazon Web Services (AWS),” Online at http://aws. amazon.com, 2008.

**[4]** Q. Wang, K. Ren, W. Lou, and Y. Zhang, “Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance,” *Proc. of IEEE* *INFOCOM*, 2009.

**[5]** End-User Privacy in Human–Computer Interaction. http://www.cs.cmu.edu/~jasonh/publications/fnt-end-user-privacy-in-human-computer-interaction-final.pdf.

**[6]** D. Song, D. Wagner, and A. Perrig. Practical techniques for searching on encrypted data. In IEEE Symposium on Research in Security and Privacy, pages 44{55. IEEE Computer Society, 2000.

**[7]** J. Heiser and M. Nicolett, “Assessing the security risks of cloud computing,” *Gartner Report*, 2009. [Online]. Available: http://www.gartner.com/DisplayDocument? id=685308