COLLABORATIVE APPROACH FOR EFFICIENT AUTHENTICATION, DATA AUDITING, AND DATA AVAILABILITY IN CLOUD COMPUTING

Ph.D. Thesis

By

RAJAT SAXENA

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MARCH 2019

COLLABORATIVE APPROACH FOR EFFICIENT AUTHENTICATION, DATA AUDITING, AND DATA AVAILABILITY IN CLOUD COMPUTING

A THESIS

submitted to the

INDIAN INSTITUTE OF TECHNOLOGY INDORE

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

DOCTOR OF PHILOSOPHY

by

RAJAT SAXENA

DISCIPLINE OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY INDORE

MARCH 2019

INDIAN INSTITUTE OF TECHNOLOGY INDORE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled **Collaborative Approach for Efficient Authentication, Data Auditing, and Data Availability in Cloud Computing** in the partial fulfillment of the requirements for the award of the degree of **Doctor of Philosophy** and submitted in the **Discipline of Computer Science and Engineering, Indian Institute of Technology Indore,** is an authentic record of my own work carried out during the time period from January 2012 to March 2019 under the supervision of Dr. Somnath Dey, Assistant Professor, Indian Institute of Technology Indore, Indore, India.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other institute.

Signature of the Student with Date

(Rajat Saxena)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Signature of Thesis Supervisor with Date

(Dr. Somnath Dey)

Rajat Saxena has successfully given his Ph.D. Oral Examination held on _____

Signature of Chairperson, OEB	Signature of External Examiner	Signature of Thesis Supervisor
Date:	Date	Date:
Signature of PSPC Member #1	Signature of PSPC Member #2	Signature of Convener, DPGC
Date:	Date	Date:

Signature of Head of Discipline

Date:

ACKNOWLEDGEMENTS

I wish to seize this opportunity to acknowledge all who have assisted, one way or another, to the accomplishment of this dissertation, which marks another important milestone in my life.

First and foremost, the sincerest gratitude ascribes to my supervisor Dr. Somnath Dey for his immense support and unrelenting source of motivation throughout my PhD tenure. His benevolent supervision, constructive criticism, invaluable guidance, and financial support from the moment I started working at IIT Indore as a doctoral student. The consistent and unending directions steered me in the right path and transformed me into a researcher who can work independently. I could not have imagined having a better, affable supervisor and mentor for my doctoral tenure.

Besides, I extend my gratitude to my research progress committee members: Dr. Aruna Tiwari and Dr. Santosh Kumar Vishvakarma, for their keen observation, valuable comments, insightful suggestions, encouragement, and validation of this research. I would like to express my appreciation to Dr. Surya Prakash, Head of Discipline for all his extended suggestions and support.

I express sincere gratitude to Prof. Pradeep Mathur (Director, IIT Indore), who has been highly encouraging during the entire course of my doctoral work. I also convey sincere thanks to Dr. Abhishek Srivastava, Dr. Neminath Hubbali, Dr. Aruna Tiwari (DPGC Convener, Discipline of Computer Science and Engineering) for their support and motivation all the times. A special gratitude goes to Indian Institute of Technology Indore for providing me an opportunity to pursue PhD under state-of-the-art research environment and also Ministry of Human Resource Development (MHRD), Government of India for granting financial aid to carry out this doctoral study successfully.

I am also grateful to cheerful doctoral fellows; Dheeraj, Raj Kumar, Rudresh, Ram Prakash, as well as the undergraduate students for their feedback, encouragement, cooperation, and of course friendship. Without their precious support, it would not be possible for me to reach this level. Also, I would wish to thank the all staffs of Computer Science and Engineering Discipline.

Finally, I must express my profound gratitude to my family: my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of doctoral research and writing this thesis. Their belief in me and my capabilities is the prime motivation that has driven me to this stage where I can confidently take decisions to realize my dreams. This accomplishment would not have been possible without them.

_____

Rajat Saxena

Dedicated to My Parents

ABSTRACT

Cloud computing is the next-generation computing technology which provides different "on-demand" services by dynamically sharing resources among multiple users over Internet. This technology can easily be adopted by novice end users. However, various security threats have been introduced into the CIA (confidentiality, integrity, and availability) triad of cloud. To manage these threats, two types of approaches have been followed. The first approach uses isolation of network resources, users, and applications and the second approach ensures secure software design, and formal and strict testing procedures. Recent experience shows that both approaches are irrelevant to the cloud deployment modules due to their adequate level of security. In cloud environment, end users require guarantee of security for the deployment of reliable and trusted network applications, which are maintained by Cloud Service Provider (CSP). Therefore, we need a novel approach that enforces secure network application and environment for cloud users.

In this work, first, we propose a solution for secure and efficient authentication mechanism in cloud environment. In our proposed methodology, user's data are stored into various cloud servers, and a bilinear pairing based secret key generation method is proposed to provide better security. A Secret Key Generator (SKG) is utilized to keep the identities for users and to perform authentication in cloud. From the stored identities, a hierarchy is created to generate the secret key for a user. In each hierarchy level, an intermediate key is generated from the user's identity stored corresponding to that hierarchy level and the intermediate key of the previous level. This process uses bilinear pairing to generate the secret key. Further, SKG generates different public parameters to perform authentication of user with different CSPs. The proposed mechanism is able to prevent malicious users from accessing legitimate resources.

Next, we propose two different techniques for data integrity verification (DIV). The proposed techniques utilize multiple third-party auditors to reduce the computational overhead of the end user. In both approaches, a file is divided into a number of blocks before storing into different cloud service providers. Then, algebraic signature (AS), combinatorial batch codes (CBC), and homomorphic tag are used in the first approach; whereas Paillier homomorphic cryptography (PHC) system, and CBC are used in the second approach. AS is used to generate homomorphic code and CBC is utilized to store file blocks into different CSPs. The AS helps to maintain confidentiality of the data during the integrity check using TPA

in the first approach. In the second approach, encrypted file blocks are stored into different CSPs using CBC. The PHC helps to preserve the confidentiality of the data. Moreover, the properties of AS and PHC support dynamic operations for DIV method 1 and 2, respectively.

Finally, a collaborative model using Multiple Third Party Auditors (M-TPA) is proposed for DDoS attack prevention. Here, we utilize Dempster Shafer Theory (DST) and Weibull Probability Distribution to analyze the traffic pattern of the CSPs. We also compute basic probability assignment (BPA) for TCP, ICMP, and UDP packets. Based on these assessments, we detect whether a DDoS attack is occurred or not. In this model, we follow packet trace back method to identify the source of attack.

We have evaluated the proposed models with respect to different performance parameters. We achieve an average user and CSP performance of 91.41% and 96.49%, respectively for the proposed efficient authentication mechanism. Further, we attain an average accuracy of 95% and 92.76% for CBC based and PHC system based data integrity verification methods, respectively. Both the DIV approaches support dynamic data operations with adaptable and useful batch auditing through which various audit sessions for multiple users can be handled simultaneously. Finally, high sensitivity, specificity, and accuracy with a low false alarm rate are observed for the proposed DDoS attack prevention mechanism.

PUBLICATIONS FROM THE THESIS

The following mentioned publications have evolved from this doctoral dissertation (as of August 2019):

Papers in Peer - Reviewed Journals:

Published / Accepted

1. R. Saxena, and S. Dey, "DDoS Prevention using Third Party Auditor in Cloud Computing," *Iran Journal of Computer Science: An International Journal*, Springer. (Online First: 12 June 2019).[https://doi.org/10.1007/s42044-019-00039-w]
2. R. Saxena, and S. Dey, "Data Integrity Verification: A Novel Approach for Cloud Computing," *Sadhana*, vol. 44, no. 3, pp. 44-74, 2019. [SCI Expanded, IF:0.769].
3. R. Saxena, and S. Dey, "A Generic Approach for Integrity Verification of Big Data," *Cluster Computing : The Journal of Networks, Software Tools and Applications*, vol. 22, no. 2, pp. 529-540, 2019. [SCI Expanded, IF:1.851].
4. R. Saxena, and S. Dey, "Light Weight Access Control Mechanism for Mobile-based Cloud Data Storage," *International Journal of Next-Generation Computing*, vol. 9, no. 2, pp. 340-351, 2018. [Emerging SCI]
5. R. Saxena, and S. Dey, "On-demand Integrity Verification Technique for Cloud Data Storage," *International Journal of Next-Generation Computing*, vol. 9, no. 1, pp. 33-50, 2018. (Emerging SCI)
6. R. Saxena, and S. Dey,"A Curious Collaborative Approach for Data Integrity Verification in Cloud Computing," *CSI Transactions on ICT*, vol. 5, no. 4, pp. 407-418, 2017.

Submitted:

1. R. Saxena, and S. Dey, "Third Party Auditor based Data Integrity Verification in Cloud Computing," Service Oriented Computing and Applications, Springer.

Papers in Refereed Conferences

1. R. Saxena, and S. Dey, "Cloud Audit: A Data Integrity Verification Approach for Cloud Computing," in *Proceedings of the* 12^{th} *International Conference on Communication Networks (ICCN-2016)*, Bengaluru, India (also appeared in Elsevier Journal Procedia Computer Science, vol. 89, pp. 142-152, 2016). (**Best Paper Award**).
2. R. Saxena, and S. Dey, "A Novel Access Control Model for Cloud Computing," in *Proceedings of the* 9^{th} *International Conference on Internet and Distributed Computing Systems-(IDCS-2016)*, Wuhan, China, vol. 9864, Springer International Publishing Switzerland, pp. 81-94, 2016.
3. R. Saxena, and S. Dey, "Cloud shield: Effective Solution for DDoS in Cloud," in *Proceedings of the* 8^{th} *International Conference on Internet and Distributed Computing Systems-(IDCS-2015)*, Windsor, UK, vol. 9258, Springer International Publishing Switzerland, pp. 3-10, 2015. (**Best Student Paper Award**).
4. R. Saxena, and S. Dey, "Collaborative Approach for Data Integrity Verification in Cloud Computing," in *Proceedings of the* 2^{nd} *International Conference on Security in Computer Networks and Distributed Systems (SNDS-2014)*, Trivandrum, India, vol. 420, Springer, Berlin Heidelberg, pp. 1-15, 2014. (**Best Paper Award**).

Book

1. R. Saxena , *Explore Data Integrity Verification in Cloud Computing,* Lambert Academic Publishing (LAP), Germany, ISBN (13): 978-3-330-04843-0, 2017.

Book Chapter

1. S. Ruj, and R. Saxena, "Securing Cloud Data," in book title: *Cloud Computing with e-science Applications*, CRC Press/Taylor & Francis, USA, ISBN (13): 978-1-4665-9116-5, 2015.

Accepted Abstract

1. R. Saxena, and S. Dey , "A Novel Authentication Approach for Cloud Computing," in *Proceedings of International Conference on Recent Trends in Engineering and Material Sciences (ICEMS-2016)*, Jaipur, India, pp. 1721-1722, 2016.

Contents

ABSTRACT **i**

LIST OF PUBLICATIONS **iii**

TABLE OF CONTENTS **v**

LIST OF FIGURES **x**

LIST OF TABLES **xii**

LIST OF ABBREVIATIONS & ACRONYMS **xiii**

1 Introduction **1**

1.1 Basics of Cloud Computing 1

1.2 Characteristics of Cloud Computing 4

1.3 Cloud Security & Threats in Cloud Computing 5

1.3.1 Cloud Security 6

1.3.2 Security Threats 6

1.4 Motivations 7

1.5 Objectives 9

1.6 Summary of Contributions 10

1.6.1 Effective Authentication 10

1.6.2 Data Integrity Verification 10

1.6.3 DDoS Attack Preservation 11

1.7 Organization of the Thesis 11

2 Related Work **13**

2.1 Efficient Authentication 13

2.2 Data Integrity Verification (DIV) 16

2.3 DDoS Attack Prevention 22

2.3.1 Types of DDoS Attack 22

2.3.2 DDoS Attack Tools 23

2.3.3 DDoS Defense Mechanism 25

2.4 Summary 29

3 Efficient Authentication **31**

3.1 Notations and Preliminaries 31

3.1.1 Basic Notations 32

3.1.2 Bilinear Pairing 32

3.2 Proposed Technique 33

3.2.1 System Model 33

3.2.2 Workflow of the Proposed Scheme 34

3.3 Security Analysis 40

3.4 Summary 43

4 Data Integrity Verification **45**

4.1 Notations and Preliminaries 45

4.1.1 Basic Notations 46

4.1.2 Algebraic Signature 46

4.1.3 Homomorphic Tag (HT) 48

4.1.4 Combinatorial Batch Codes (CBC) 48

4.1.5 Paillier Homomorphic Cryptography (PHC) System 51

4.1.6 Some Useful Theorems 52

4.2 DIV Method 1: Using Algebraic Signature, CBC, and Homomorphic Tag . 53

4.2.1 System Model 53

4.2.2 Proposed Technique 54

4.2.3 Dynamic Data Operations 58

4.2.4 Proof of Batch Auditing 62

4.2.5 Efficiency Improvement 65
4.3 DIV Method 2: Using PHC System and CBC 65
4.3.1 System Model 65
4.3.2 Workflow of the Proposed Method 66
4.3.3 Correctness Proof of Privacy Preservation 69
4.3.4 Support for Dynamic Data Operations 71
4.3.5 Use Case 73
4.4 Experiments 75
4.5 Summary 75

5 Distributed Denial of Service (DDoS) Attack Prevention 77
5.1 Preliminaries 78
5.1.1 Dempster Shafer Theory (DST) 78
5.1.2 Weibull Probability Distribution 79
5.2 Proposed System Model 81
5.3 Proposed Technique 83
5.4 Summary 87

6 Experimental Results 89
6.1 Experiment Results for Authentication Method 89
6.1.1 Experimental Setup 89
6.1.2 Parameters 92
6.1.3 Performance Analysis 97
6.2 Experimental Results of Data Integrity Verification Methods 99
6.2.1 Experimental Setup 100
6.2.2 Parameters 100
6.2.3 Performance Analysis of DIV Methods 102
6.3 Experiment Results of DDoS Attack Prevention Method 111
6.3.1 Parameters 111
6.3.2 Test Plan 112
6.3.3 Test Model 113
6.3.4 Port Analysis 115

6.3.5 Performance Analysis 117
6.4 Summary 118

7 Conclusion and Future Work **119**
7.1 Effective Authentication 119
7.2 Data Integrity Verification 121
7.3 DDoS Attack Preservation 122
7.4 Future Research Direction 123

Bibliography **124**

List of Figures

1.1 Cloud Computing Model 2
1.2 Security Threats Associated with Cloud Computing Model 7

2.1 Life Cycle of DDoS Defence System 25

3.1 Workflow of the Proposed Method 35
3.2 Secret Key Generation Process 38

4.1 Workflow of the Proposed DIV Method 1 55
4.2 Dynamic Data Operations in DIV Method 1 60
4.3 Batch Auditing Process in DIV Method 1 63
4.4 Workflow of the Proposed Privacy Auditing DIV Method 2 66
4.5 Dynamic Update and Append Operation in DIV Method 2 74
4.6 Dynamic Delete Operation in DIV Method 2 74

5.1 Bathtub Curve of Failure Rate 81
5.2 System Model of the Proposed Method 82
5.3 Cloud Warrior Architecture 83
5.4 Probability Extraction Tree for VM V_1 84

6.1 Phases of Our Authentication Method for Cloud Environment 91
6.2 Comparative Analysis of Average User Performance 99
6.3 Comparative Analysis of Probability of Server Misbehaviour Detection for DIV Methods 103
6.4 Comparative Results of Accuracy Parameter for DIV Methods 105
6.5 Comparative Results of Availability Parameter for DIV Methods 106

6.6 Performance Comparison of DIV Methods 106
6.7 Verification Delay for the Proposed DIV Methods 107
6.8 Verification Overhead for the Proposed DIV Method 1 108
6.9 Verification Overhead for the Proposed DIV Method 2 108
6.10 Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different File Size 109
6.11 Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Detection Probabilities 109
6.12 Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Audit Frequency 110
6.13 Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Sampling Ratio 110
6.14 Test Model of DDoS Attack Prevention 114
6.15 Rate of Packets in Different Destination Port (a) TCP (b) UDP 116
6.16 Traffic Behavior (a) Before Attack (b) After Attack 117

7.1 Results of the proposed TPA based DDoS Attack Prevention Method with Respect to Different Parameters 123

List of Tables

2.1 Comparative Analysis of different Authentication Methods 16
2.2 Comparison of Different PDP Schemes 18
2.3 Comparison of Different PoR Schemes 20
2.4 Comparison of Different Data Auditing Techniques with TPA 21
2.5 Comparative Analysis of Different DDoS Attack Tools 24
2.6 Comparative Analysis of Different DDoS Defense Schemes 27

5.1 Boolean Truth Table for the OR Gate 79
5.2 Three-valued Logic Result for Four Virtual Nodes 86

6.1 Assessment of Fake Request Rate 93
6.2 Assessment of Unauthorized Request Rate 93
6.3 Assessment of Resource Affection Rate 94
6.4 Assessment of Turn Around Efficiency 95
6.5 Assessment of Resource Availability 96
6.6 Assessment of Successful Transaction Rate 96
6.7 Assessment of Integrity Preservation 97
6.8 Assessment of User Performance 98
6.9 Assessment of CSP Performance 100
6.10 Accuracy Assessment of Our DIV Method 1 104
6.11 Parameters Assessment of Our DIV Method 2 104
6.12 The Number of Arrival Packets on Date 30 November, 2016 114
6.13 Parameters Assessment of Our Approach 118

7.1 Assessment of User's Performance 120

7.2 Assessment of CSP Performance 121
7.3 Accuracy Assessment of Our DIV Method 1 122
7.4 Parameters Assessment of Our DIV Method 2 122

List of Abbreviations & Acronyms

CSP	Cloud Service Provider
DaaS	Data as a Service
DBMS	Data Base Management System
RDBMS	Relational Database Management System
IaaS	Infrastructure as a Service
VMs	Virtual Machines
PaaS	Platform as a Service
HT	Homomorphic Tag
SLC	Software Life Cycle
ASP	Application Service Provider
SaaS	Software as a Service
OS	Operating System
MITM	Man in the Middle
NIC	Network Interface Card
DNS	Domain Name Server
IP	Internet Protocol
DNSSec	Domain Name Server Security Extension
LAN	Local Area Network
WAN	Wide Area Network
HTTP	Hypertext Transfer Protocol
DoS	Denial of Service
DDOS	Distributed Denial of Service
VMM	Virtual Machine Manager
M-TPA	Multiple Third Party Auditors

TPA	Third Party Auditor
CBC	Combinatorial Batch Codes
DST	Dempster-Shafer Theory
HDFS	Hadoop Distributed File System
HIBE	Hierarchical Identity-Based Encryption
SDP	Software Development Process
UDP	User Datagram Protocol
TCP	Transmission Control Protocol
ICMP	Internet Control Message Protocol
BPAs	Basic Probability Assessments
MTTF	Mean Time to Failure
TTL	Time To Live
UFID	Unique File Identifier
PRF	Pseudo-Random Function
PRP	Pseudo Random Permutation

Chapter 1

Introduction

Cloud Computing introduces a wide range of advantages including elasticity of computing resources, economic savings, and service flexibility. However, security and privacy concerns are the major challenges to a wide adoption of cloud computing. In this thesis work, we are going to propose solutions for efficient authentication, data integrity verification, and prevention of DDoS attacks. Section 1.1 of this chapter gives the basic concept of cloud computing along with the different service model. Section 1.2 provides the characteristics as well as the advantages of cloud computing. We discuss the security threats concerning to the cloud computing model in section 1.3. In section 1.4, we present the motivation of this work. Section 1.5 states the objectives of our work. In section 1.6, we highlight the major contributions made in this thesis. Finally, organization of the thesis is given in section 1.7.

1.1 Basics of Cloud Computing

Cloud computing is an exciting and promising technology paradigm that changes current technological and computing concepts into utility-like solutions. It is the "on-demand" delivery of computing services [1–3] like servers, database storage, applications, networking, software, analytics, intelligence, and more IT resources via the Internet ("the cloud services platform") to provide rapid access to flexible resources. In cloud, users typically pay only for cloud services they use which help to manage infrastructure more efficiently with low cost. It provides a simple way to access servers, storage, databases, and a broad range of application services over the Internet.

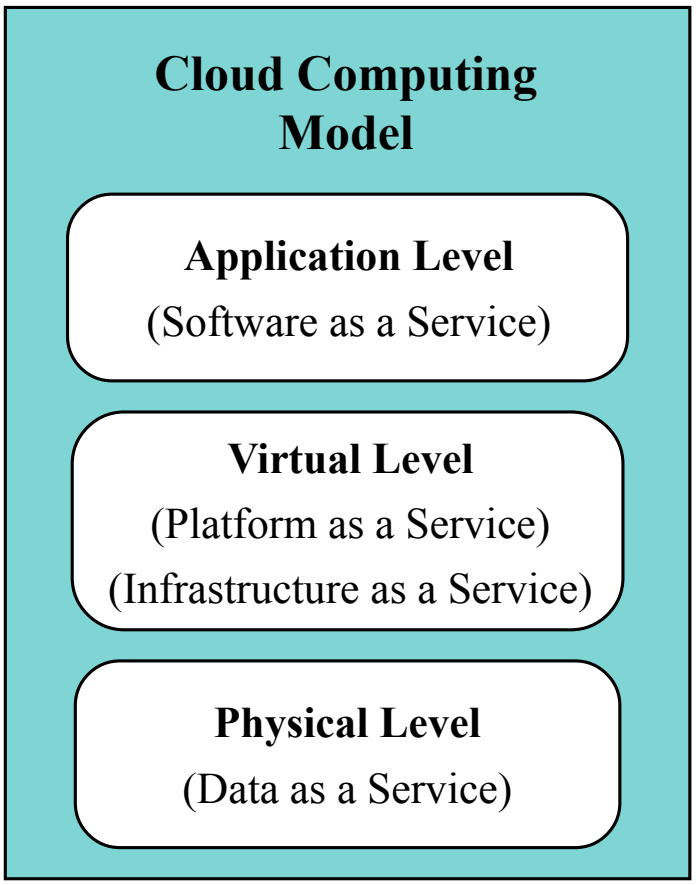


Figure 1.1: Cloud Computing Model

Service Models of Cloud Computing

Cloud computing provides different services depending on users' requirements. We model these services in three different levels as shown in Fig. 1.1. These service levels are briefly discussed in the following.

- **Physical Level:** The data storage is one of the main priority of any service gathering. If the user chooses the easy option for data saving at the end of a service provider, storage server of the service provider must be available with the facility of Data as a Service (DaaS).

  DaaS is one type of infrastructure service which delivers on-demand virtual data storage to fulfill user requirements [4]. It allows users to pay only what they used rather than entire license subscription fee for Data Base Management System (DBMS). It offers an extensive amount of data storage with a compressed format that is designed to cope up with large, inexpensive, and fast data storage. It also supports traditional

storage system such as Relational Database Management System (RDBMS) and file systems with new commercial technologies.

The examples of DaaS: - Amazon S3, Google BigTable, and Apache HBase.

- **Virtual Level:** Virtualization is the elementary automation of resource allocation and supply to accomplish user's request. The virtual resources like infrastructure and platform are required to improve the quality of advancement and implementation of rigid technologies. At virtual level the following two services are achieved by users.

  1. **Infrastructure as a Service (IaaS):** This service model contributes virtual resources (applications, storage, networks, and operating systems, etc.) for computing over the Internet [2]. IaaS uses virtualization as a core for integration or decomposition of physical resources to qualify for increasing or decreasing demands invoked by the users. The primary goal of virtualization is the setup of separate Virtual Machines (VMs) for service provision, which is isolated from basic hardware and other VMs. This approach is different from multi-tenancy that transforms the multiple instances of multiple users to run on a single application.

     The examples of IaaS: - Amazon's EC2, AWS Elastic Beanstalk, Windows Azure, Heroku, Force.com, Google App Engine, Apache Stratos, Office365.

  2. **Platform as a Service (PaaS):** This service model provides a precise framework, essential functions, and Software Life Cycle (SLC) to develop, build, test, and host applications, which can be customized and employed by the user to develop their applications [3]. It also authorizes the user to expand application and services directly on the cloud.

     The examples of PaaS: - Amazon EC2, Windows Azure, Rackspace, Google Compute Engine.

- **Application Level:** The Cloud is an extension of Application Service Provider (ASP). It maintains high application availability with the common users to fulfill their online demand. The Software as a Service (SaaS) is fundamental service provision at the application level. With SaaS, users can access different hosted software application on PaaS through networks [5]. SaaS applications are built by the service provider and could be configured by users. However, the user does not have rights to change or

modify this request. The cost of using this application is nominal; even less than a software subscription charges. The SaaS application requires optimization regarding availability, disaster recovery, speed, security, and maintenance.

The examples of SaaS: - SalesForce.com, OneDrive Sync, Google Docs, Google Apps, Microsoft Office 365, Workday, Concur, Citrix GoToMeeting, Cisco WebEx, etc.

1.2 Characteristics of Cloud Computing

Cloud computing has a variety of characteristics. Some important characteristics of the cloud are explained below:

1. **Resource Pooling:** Computing resources (system memory, cache memory, hard disk space, Interrupt Requests (IRQs), and Direct Memory Access (DMA) channels) are used by multi-tenant model for the provision, in which multiple customers pool their required resources and fulfil the constant demand of virtual and real resources. The beautiful thing about it is that the user does not aware of the exact location of the resources.
2. **Virtualization:** The concept of virtualization allows multiple and potentially varied operating system instances to run concurrently on a single physical computer system. Each of these instances shares the physical resources (Memory, Storage, CPU, and Network Connectivity) of the host computer system. With the help of virtualization, a single machine can provide an IT infrastructure that could require multiple computer systems. Thus, the user is able to perform those operations that cannot be possible through a single computer.
3. **Elasticity with Ultra Large Scale:** Elasticity is the ability of any system to resist on a distortion influence and yield to its previous original condition, when this distortion influence is removed. In cloud, the virtual resources have dynamic and rapid elasticity in case of increasing and decreasing demand for the resources. It provides a flexible choice for the user to select unlimited quantity and variety of resources at any time. The capability of scaling [6] in cloud environment is very significant.
4. **On-Demand Self-Service:** Any user can automatically acquire the demanded resources without any latency delay and human interaction. For this, Cloud Service

Provider (CSP) provisioned the metered computation capabilities, such as network storage and server as per the requirements of the users.

5. **Broad Network Access:** The cloud provides a wide variety and range in standard mechanism (Hypertext Transfer Protocol Secure (HTTPS), Secure Sockets Layer (SSL), Secure Shell (SSH), and Secure Remote Login (SRL)) for network access of different utilities (laptops, mobiles, and tablets, etc.).
6. **Reliability:** Cloud services are more reliable [7] than other traditional approaches because it uses multi-transcript fault-tolerant methods for handling any situation of failure. It avoids the cases of single point failure.
7. **Inexpensive:** Cloud computing is an inexpensive technology because of its metered services with pay-as-per-use model and user does not need to invest for his own infrastructure. Cloud fault tolerance mechanism is expandable within a large number of economic nodes. Due to this reason, scaling capability of resources is high with the minimum failure rate. Thus, users can take advantage of low-cost resources.
8. **Backup and Disaster Recovery:** The Cloud provides rapid and automatic disaster recovery capabilities and backup service for any platform. It is very useful for the small companies in which we find the lack of technical expertise and required expenses.
9. **Automatic Software Updates:** Cloud servers are off-premise, out of sight and out of user control. Thus, it is the responsibility of CSP to update regularly and integrate software for their provided services. It is based on the strict compliance and regulatory obligations of standards set by auditing authorities.
10. **Quick Deployment:** The Cloud technology offers a very vast variety of implementation services. Once you choose a model type, out of public, private, commodity or hybrid, then, in the few minutes, entire systems would be fully functional for obtaining your giant business goal.

1.3 Cloud Security & Threats in Cloud Computing

In this section, first we discuss the security aspect of cloud computing. Thereafter, we will present the different security threats associated with each level of the service model.

1.3.1 Cloud Security

To adapt a new technology with confidence, security is the primary requirement of any organization. Security is the assurance of any independent entity to approach a measurable goal besides the threats and dangers generated by the intruders. Security can be reformed as the composite fusion scripts of three fundamental qualitative entities.

1. **Confidentiality:** It is the avoidance of unauthorized exposure of the information. It also involves various rules and restrictions that limit access to certain information and places.
2. **Integrity:** It refers to the avoidance of unauthorized alteration, modification, or deletion of the information. It is the quality of the system to be persistent during any change occurred in the system.
3. **Availability:** It is the avoidance of illegal prohibition of the information. It is the degree of the system to be committed for operable at any specific instant of time.

1.3.2 Security Threats

In the cloud, there are a vast variety of security threats and hazards that may damage the trust of a user on service providers. The user must have confidence in the service providers for taking proper security arrangement in case of any threatening conditions. Virtualization is one of the key points to attack in the cloud environment, because of the vulnerable relationship between hardware and operating system (OS). In this condition, virtualization software "hypervisor" may be compromised. Thus, cloud environment must support multi-tenant and isolation for prevention of the security threats. The availability, integrity, confidentiality, and reliability of resources can be severely affected due to many attacks. Thus, avoidance of security threats must be the primary concern in the cloud computing.

We have listed different security threats associated with each level of the cloud computing model. These threats are shown in Fig. 1.2.

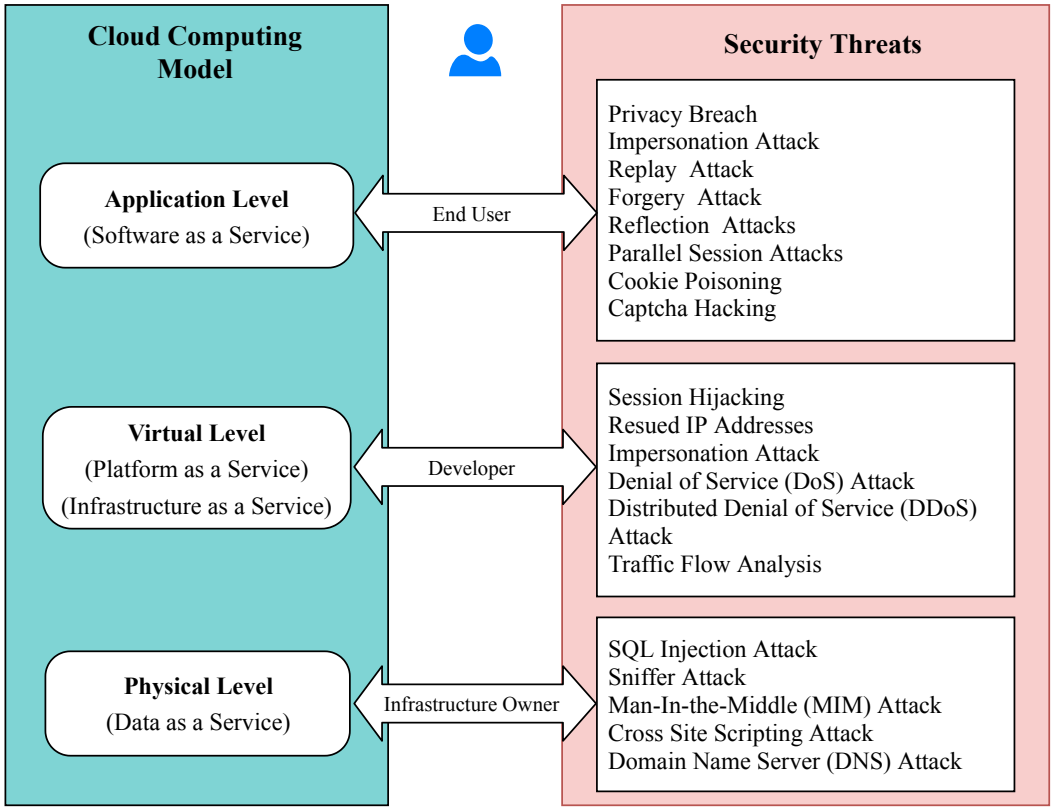


Figure 1.2: Security Threats Associated with Cloud Computing Model

1.4 Motivations

The growth in cloud computing services, however, has made it vulnerable to various security threats for novice cloud users. Major vulnerable attacks in cloud computing are performed to breach the authentication, data integrity and availability of services [8–10].

In the cloud environment, authentication is a major concern especially during the deployment of cloud computing services. It can prevent legitimate users from accessing the content stored in the cloud and permit attackers to misuse the content. Although various schemes are available to tackle attacks like impersonation attacks, replay attacks, forgery attacks, reflection attacks, and parallel session attacks [8, 11, 12] for authentication, the need for a more robust authentication mechanism is strongly felt.

The integrity of the client's data is another major concern in cloud computing because cloud service providers may not always be trustworthy. Different attacks on physical level (See Fig. 1.2) may cause data leakage which lead to loss of user data. There are mainly

two approaches proposed in literature, namely Proof of Retrievability (PoR) [9, 13, 14] and Provable Data Possession (PDP) [15–17] to verify data integrity. However, the verification process of these techniques depends on a two-tier architecture and they can only verify static data efficiently. The issue is their inefficacy in performing dynamic operations. Dynamic operations are required when a user wants to modify only a few data blocks in the original stored data. Existing approaches maintain a trade-off between dynamic operations, communication complexity, and storage. Certain other techniques [18–20] comprise a three-tier architecture, that involves trusted Third Party Auditors (TPA) that serve as an interface between the cloud user and the CSP. The major issues with these techniques are: single point failure and inefficient batch auditing.

A third significant concern in cloud computing is the availability of services. The availability of services may be adversely affected by the session hijack attacks, impersonation attacks, reused IP address attacks, and Distributed Denial of Service (DDoS) attacks. Among these attacks, DDoS is a collaborative attack on the availability and functionality of a victim cloud through multiple corrupted systems. Subsequent to stealing control, an associate intruder compels these compromised systems to send a large number of malicious packets towards the victim cloud. Various solutions for DDoS attack have been proposed in literature. A few of them [10, 21–24] are based on intrusion detection systems (IDS). However, such DDoS prevention techniques are affected by impersonation attacks, privacy leaks, poor performance, and the issue of single point failure.

This thesis focuses on providing an effective solution leading to an authentication mechanism that can prevent malicious users from accessing legitimate resources and provide better security. Further, to resolve the data integrity verification problem, we explore Multiple TPA based solutions. Further, we investigate the applicability of CBC, Algebraic Signature [25, 26], and Homomorphic Tag for data integrity verification. We also examine the possibility of data integrity verification with CBC and a variant of the Pailliar cryptography system. Finally, we investigate for a solution to prevent DDoS attacks using TPA and Weibull Distribution in cloud environment.

To avoid these threats, several solutions are available. In literature, only few works use a trusted third party for evaluation of message communication or exchange. It works as a fair interface between users and service providers. However, frequent message communication

increases time and space complexity at the end of service provider.

1.5 Objectives

Performance and security are two major concerns in the wide deployment of the cloud computing systems. Based on the context described above, the objectives pursued in our work are divided into three broad categories. These objectives are stated in the following:

- **Objective 1: Effective Authentication**
  - To investigate and improve third party collaboration for secure authentication mechanisms in cloud computing.
  - To make use of hierarchical bilinear pairing for effective and better authentication in the cloud environment.
  - A comparative analysis with traditional authentication methods to prove the effectiveness of the proposed approach.
  - To identify the most suitable and efficient collaboration mechanism between third party auditors (TPA), cloud service providers (CSP), and cloud users to find a common security goal.
- **Objective 2: Data Integrity Verification**
  - To propose an efficient data integrity verification mechanism with multiple third party auditors (M-TPA) for work collaboration in the cloud environment.
  - To make use of Combinatorial Batch Codes (CBC) for load balancing among various CSPs to perform dynamic data operations in the cloud.
  - To explore probabilistic methods for data integrity verification which could reduce the number of verification challenges.
  - To support batch auditing with the implementation of applications in the Hadoop and MapReduce framework.
- **Objective 3: DDoS Attack Prevention and Mitigation**
  - To propose prevention mechanisms for DDoS attacks in cloud environments with the help of third party packet trace back technique.
  - To make use of Dempster-Shafer Theory (DST) and Weibull probability distribution for analyzing the impact and source of DDoS flood attacks and to prevent

such attacks.

– To analyze the security and performance of our DDoS prevention techniques based on small size plain text.

1.6 Summary of Contributions

In this work, we address several security concerns associated with cloud computing and enhance the current state of the work in the manner listed below. Multiple third party auditors (M-TPA) work as an interface between CSPs and cloud users, and help them to efficiently perform the following tasks in the scenarios of limited availability of resources.

- **Effective authentication.**
- **Data integrity verification.**
- **DDoS attack prevention.**

In the following subsections, we present the significant contributions related to each task.

1.6.1 Effective Authentication

In this research work, we have explored the scope of TPA for authentication. We propose a novel authentication method for cloud environments, in which a secret key for the user is generated through a hierarchical development of bi-linear mapping. We have implemented a prototype of the proposed method in the Hadoop and MapReduce framework and evaluated the user CSP performances with different parameters under varied conditions. We have also done a comparative analysis of our scheme with other methods. From the obtained results, it is evident that our scheme provides more effective solution for cloud environments.

1.6.2 Data Integrity Verification

In the second part of the work, we propose two collaborative techniques for data integrity verification with multiple third-party auditors. The first technique utilises the algebraic signature, homomorphic verification tag, and combinatorial batch codes while the second technique utilises the Paillier homomorphic cryptography system for data integrity verification. Properties of algebraic signature and CBC demonstrate the suitability for data integrity ver-

ification in cloud computing. On the other hand, the properties of the Paillier homomorphic cryptography system make the second approach very helpful in performing data integrity verification in a secure manner in cloud environments. Both approaches achieve high probability of server misbehavior detection with good accuracy. Further, both the approaches support dynamic operations with less overhead. Performance comparison shows the effectiveness and utility of these approaches.

1.6.3 DDoS Attack Preservation

Finally, we have proposed a collaborative approach for DDoS detection and prevention based on third party auditors. This method uses DST and Weibull distribution for DDoS detection and prevention. Three valued logic value of Weibull distribution makes it ideally suited for cloud storage. The Security service model of our approach ensures the security of the CSPs. We have also addressed the issue of IP spoofing in this work. Further, the experimental evaluation of the proposed approach shows that the proposed method successfully handles DDoS attack in cloud environments.

1.7 Organization of the Thesis

The rest of the thesis is organized in the following chronicle chapters.

- In chapter 2, we survey and analyze existing work related to efficient authentication, data integrity verification, and DDOS attack prevention methods.
- Chapter 3 presents the proposed efficient authentication mechanism in cloud environments.
- In chapter 4, first, we describe a TPA-based novel data integrity verification (DIV) technique that uses algebraic signature, CBC, and homomorphic tag. The latter part of the chapter discusses another variant of this technique using the Paillier cryptography system.
- Chapter 5 presents a new packet trace back technique for DDoS attack prevention. This technique utilizes the trusted TPA for packet analysis on behalf of users.
- In chapter 6, we report the performance parameters as well as the experimental results obtained in our work.

- Finally, chapter 7 concludes our work. We also discuss future research directions evolved during this work.

Chapter 2

Related Work

Important research papers brought a significant technical development in the last few years and this chapter presents the survey of those technical developments. Section 2.1 reviews various authentication methods for cloud computing. This section also includes merits and demerits of these methods. In section 2.2, the works related to DIV approach are analyzed. Different DDoS attack prevention techniques are discussed with their limitations in section 2.3. Section 2.4 summarizes this chapter.

2.1 Efficient Authentication

In this section, we survey the literature related to the existing authentication methods in the cloud environment. First, we discuss some basic authentication methods, which are also used in cloud environment. Then, we describe existing methods, which are specifically built for cloud environment.

Do et al. [27] showed that to authenticate with the Dropbox [28], a user requires the OAuth token, OAuth consumer key, and the OAuth signature. It is also observed that any adversary can retrieve all the confidential information from the Dropbox directory except that half of the OAuth signature. This is a big breakthrough for compromising the commercial cloud authentication to fetch any confidential file, adversary requires OAuth consumer key and the missing half of the OAuth signature.

Authorization Federation [29] is a central policy decision point to delegate service access request. It provides a homogeneous authorization for multi-cloud environment. Limitations

of this approach are single point failure, poor performance, and high network latency. The work proposed in [11] analyzed an efficient and scalable authentication scheme. This scheme uses the property of multi trap door hash functions for storing data from multiple sources. The drawbacks of this method are expensive pairing operations and linear growth for tag aggregation. It involves minor performance degradation in comparison to other methods.

A new Privacy Aware Authentication (PAA) scheme is proposed by He et al. [8], which is based on identity signature in mobile cloud computing. This method requires less computation and communication cost. The major disadvantage of this scheme is that it is less secure against Impersonation Attack. Jiang et al. [30] suggested a novel method, which includes three factor authentication for user of vehicular cloud computing. In this method, single sign-on token is used for flexible, scalable and secure access of different services. This method requires less computation cost and low communication overhead. However, this scheme is less secure while transmitting the data because it causes privacy leaks.

A privacy aware mutual authentication scheme is suggested by Jiang et al. [31] to tolerate all major security threats. This scheme improves the design flaws in many cases such as misuse of biometrics, wrong password, and fingerprint login. It also improves no user revocation facility for distributed mobile cloud servers. An application oriented architecture is proposed by Amin et al. [32] for distributed cloud environment. With the help of this architecture, registered users can access secure private information from cloud servers. This approach provides an informal cryptanalysis protocol, which protects user from all possible security threats. But this approach suffers from the problem that protection from security attacks is dependent upon hardness assumption of the hash function.

Benzekki et al. [33] proposed a context aware authentication system which implements conjunction with password protection. This scheme is cost effective and easy to implement. But main drawback of this method is that authentication request passes through different steps for an access decision. Wu et al. [34] proposed a two factor authentication, in which adversary can guess the password with negligible probability. They also demonstrate a formal method to crack a session key and privacy of the scheme. This scheme is only intended for smart health care systems. A provable secure authenticated scheme is proposed by Odelu et al. [12] for distributed cloud services. This scheme utilizes strong credentials privacy, which is provably secure and efficient. This scheme is shown secure against Man in the Mid-

dle Attack, Impersonation Attack, and Replay Attack. However, revelation of the session specific information is the major drawback of this scheme.

Butun et al. [35] presented a multi level secure authentication for reliable and resilient networks. It includes seamless integration of cyber physical systems and IoT based real time control. Privacy utility trade-off is the main drawback of this scheme. A secure signcryption scheme with cipher-text authentication is proposed by Daniel et al. [36]. The method is based on hardness of RSA assumption and discrete logarithm problem. This scheme is validated through automated cryptographic verification tool. The major drawback of this scheme is that it is less secure against Impersonation Attack. In [37], a trust and reputation based authentication method is proposed, which incorporates authentication to avoid malicious Impersonation Attacks. In this scheme, service of CSP is calculated in terms of trust and reputation. This scheme works suitable under collision and white washing attacks. The limited scope is the major issue of this scheme. A zero knowledge proof authentication scheme is proposed in [38]. The method is delighted over a traditional TCP/IP infrastructure. Additional bandwidth and computational requirements are the major concerns of the scheme.

A work presented in [39] described a shared authority based privacy preserving authentication method, in which anonymous access request is used for authentication. In this method, attribute based access is required to incorporate own data fields. Other contribution of work is proxy re-encryption, which is applied on user privacy. Low resistance for impersonation is the main disadvantage of this method. In other research work [40], a server side anonymous attribute based authentication scheme is proposed, which formalizes security by outsourcing the computation. Signing attributes require heavy computation overhead, which is the main limitation of this scheme.

Another work [41] presented a multi-server authentication based on biometrics and smart-card. This scheme supports many features such as anonymity, mutual authentication, and forward secrecy. This scheme provides resistance to eavesdropping attack, server spoofing attack, stolen smart-card attack, and masquerade attack. The scope of this scheme is limited to the systems, which are using smart-card. A key-aggregate authentication cryptosystem is proposed by Guo et al. [42]. The method is executed by the cloud server. In this scheme, number of keys are constant. It solves the problem of key-leakage during data sharing. However, the scheme may be suffered from collusion in the exchange of keys.

Table 2.1: Comparative Analysis of different Authentication Methods

Method	Mechanism	Advantages	Disadvantages
Do et al. [27]	User requires OAuth token OAuth consumer key, and the OAuth signature.	This is a big breakthrough for compromising the commercial cloud authentication.	The missing half of the OAuth signature requires for further processing.
He et al. [8]	Privacy Aware Authentication (PAA)	It provides a homogeneous authorization for the multi cloud environment.	This approach is suffering from single point failure, poor performance, and high network latency
Jiang et al. [30]	Three factor authentication for user of vehicular cloud computing.	Multi trap-door hash functions for storing data from multiple sources.	Expensive pairing operations and linear growth for tag aggregation.
Amin et al. [32]	An application oriented architecture for distributed cloud environment.	Less computation and communication cost.	Less secure against Impersonation Attack.
Benzekki et al. [33]	A context aware authentication system to implement conjunction with password protection.	Less computation cost and low communication overhead.	It causes privacy leaks.
Wu et al. [34]	A two factor authentication	Cost effective and easy to implement	Anyone can guess the password with negligible probability
Odelu et al. [12]	Strong credentials privacy	Secure against Man in the Middle Attack, Impersonation Attack, and Replay Attack.	Revelation of the session specific information.
Butun et al. [35]	A multi level secure authentication	Seamless integration of cyber physical systems and IoT based real time control	Privacy utility may trade-off.
Daniel et al. [36]	A secure signcryption with cipher-text authentication	Automated Cryptographic verification tool.	Less secure against Impersonation Attack.
Guo et al. [42]	A trust and reputation based authentication, incorporates authentication to avoid malicious Impersonation Attacks	Zero knowledge proof authentication scheme	Low resistance for impersonation

2.2 Data Integrity Verification (DIV)

Data auditing is a periodic event to assess quality of data for a specific purpose like to evaluate security, data integrity, privacy preservation, and computational accuracy. This could be a primary source for shielding corporate data assets against potential risk and loss. Auditing of data relies on a registry which could be a storage space for information and data assets. During data auditing, the creation, origin, and/or format of data may be reviewed to assess its utility and value.

Furthermore, DIV schemes use two types of approaches : probabilistic and deterministic. In probabilistic approach, some blocks are randomly selected to check. On the other hand, deterministic techniques [43] check the integrity of all data blocks.

Traditional systems generally use Provable Data Possession (PDP) and Proof of Retrievability (PoR) schemes for data auditing. Both of these schemes are based on the fact that

a client directly communicates with the data storage servers to produce proof of access, retrieve, and possess of the data. The only difference between PDP and PoR techniques is that PDP techniques merely produce a proof for recoverable data possession but PoR schemes check the possession of data and it can recover data in case of data access failure or data loss. Usually, a PDP scheme can be transformed into PoR scheme by adding erasure or error correcting code.

The PDP techniques [9], [13], [44], [45], [46] generate probabilistic proofs of possession by sampling random sets of blocks from the server, which drastically reduces I/O transfer costs. These techniques have two parts of actions. First, the client (verifier) preprocesses the data, and keeps a small amount of meta-data. Then, the client sends entire data to a non-trusted data storage server (prover) for storing. Later, client (verifier) verifies the original data with the help of meta-data and confirms whether the data storage server still possesses the client's original data, and the stored data has not been tampered or deleted. In PDP techniques, the client maintains a constant amount of meta-data to verify the proof. The challenge/response protocol transmits a low, constant amount of data that minimizes network communication. Hence, the PDP schemes for remote data checking support large data sets in widely distributed storage systems.

In literature, several PDP techniques have been proposed. The PDP technique proposed by Ateniese et al. [9] uses Homomorphic Verifiable Tags (HVT) as a key tool for DIV. In another work [13], they proposed a Scalable PDP technique. This work is an enhanced version of the initial PDP [9] and utilizes the symmetric key cryptography for DIV. A variant of this approach is dynamic PDP technique [45]. It uses rank-based authenticated dictionary, skip list and RSA tree for DIV. Another efficient PDP technique is proposed by Seb'e et al. [44]. The asymmetric key cryptography (RSA modules) is used as a key tool for DIV in this work. There is another PDP technique [46] which uses the algebraic signature for DIV. The different properties supported by these PDP schemes are summarized in Table 2.2.

PoR schemes [15], [16], [47], [48], [49] also have two parts of action. First, the client (verifier) allows to store a file on a non-trusted data storage server (prover). Later, the client runs data audit proof algorithm. This proof helps prover to ensure that it still possesses the client's data file, and client can recover the entire file. In these schemes, an encrypted file random embeds a set of randomly-valued check blocks or sentinels. The use of sentinel for

Table 2.2: Comparison of Different PDP Schemes

Properties	**PDP [9]**	**S-PDP [13]**	**E-PDP [44]**	**D-PDP [45]**	**C-DPDP [46]**
Primitives	Homomorphic Verifiable Tags (HVTs)	Symmetric Key Cryptography	Asymmetric Key Cryptography (RSA Modules)	Rank-based Authenticated Dictionary, RSA Tree and Skip List	Algebraic Signature
Type of Guarantee	Probabilistic	Probabilistic	Probabilistic	Probabilistic	Probabilistic
Public Verifiability	Yes	No	No	No	Yes
With the Help of TPA	No	No	No	No	No
Data Dynamics	Append only (Static)	Yes	No	Yes	Yes
Privacy Preserving	No	No	No	No	No
Support for Sampling	Yes	Yes	No	Yes	Yes
Probability of Detection	[1-$(1-p)^c$]	[1-$(1-p)^c$]	[1-$(1-p)^{c*s}$]	[1-$(1-p)^c$]	[1-$(1-p)^{c*s}$]

data auditing minimizes the client and server storage. It also minimizes the communication complexity of the audit and the number of file-blocks accessed by server during audit. An approximate executed PoR scheme encourages verifier because the prover presents a protocol interface through which the verifier can collectively retrieve the file.

There are several PoR techniques proposed in existing literature. Among these, the PoR technique proposed by Juels et al. [15] uses error correcting codes, symmetric key cryptography, sentinel creation, and permutation for DIV. PoR Hardness Amplification (PoR-HA) technique proposed by Dodies et al. [47] uses the error correcting codes, Reed-Solomon codes, and hitting sampler for DIV. In other work [48], the adversarial error correcting codes are used to solve the DIV problems. In Compact-PoR [16], the BLS signature and pseudo-random functions are used for DIV. Further, Juels et al. [49] presented a work for High

Availability and Integrity Layer (HAIL) in cloud storage. They used integrity protected error correcting universal Hash and MAC functions to solve the DIV problems. Table 2.3 summarizes the different properties of the above stated PoR schemes.

The major issue with PoR and PDP schemes is that these schemes are used only for static data, and are applied only on encrypted files which allow a limited number of queries. Further, there is a trade-off between privacy preservation and dynamic data operations. Some schemes do not preserve the privacy at all. Both PDP and PoR schemes are computation intensive and executed at the user end. None of these schemes considers batch auditing process. In addition to above issues, the effectiveness of these schemes primarily relies on the preprocessing steps which is conducted by users before out-source the data file. This also introduces significant computational and communication overhead. Furthermore, these techniques provide trade-off between storage overhead and cost of communication. Some of these techniques utilize less storage with high cost.

In cloud scenario, the clients might have limited CPU, battery power, and less communication resources. So, they may not be capable to perform data audit. In this situation, Third Party Auditors (TPA's) perform data audit on behalf of clients. Further, a trusted TPA has certain special expertise and technical capabilities, which the clients may not have.

The schemes [18], [19], [20], [50] assign auditing work to single TPA. Trusted Third Party (TTP) is an independent and trusted entity to conduct data audit. External trusted third-party audit mechanism is important, and crucial for the protection of data and the reliability of services in the cloud environment. TPAs are able to execute audit task on cloud data storage without demanding the local copy of data and do not introduce additional on-line burden to the cloud users.

The technique proposed in [18] uses Merkle Hash Tree and aggregate signature to perform the DIV. In another approach [19], bilinear map, MAC, and homomorphic authenticator are utilized for DIV. Further, RSA based bilinear homomorphic verifiable tags are also used [20] for DIV. In cooperative PDP [50], homomorphic verifiable hash index hierarchy are applied to solve the DIV problems.

Table 2.4 shows the comparative analysis of data auditing schemes which use single TPA. In these schemes, single TPA cannot handle SLA and legal issues for data possession. These techniques are also prone to single-point failure. None of the above schemes support

Table 2.3: Comparison of Different PoR Schemes

Properties	**PoR [15]**	**C-PoR [16]**	**PoR-HA [47]**	**PoR-TI [48]**	**HAIL [49]**
Primitives	Error Correcting Code, Symmetric Key Cryptography, Sentinel Creation and Permutation	BLS Signature, Pseudo-random Functions	Error Correcting Codes, Reed -Solomon Codes, Hitting Sampler	Adversarial Error Correcting Codes	Integrity Protected Error Correcting Universal Hash Function, MAC
Type of Guarantee	Probabilistic	Probabilistic	Probabilistic	Probabilistic / Deterministic	Probabilistic / Deterministic
Public Verifiability	No	Yes	Yes	Yes	Yes
With the Help of TPA	No	No	No	No	No
Data dynamics	No	No	Append only	Append only	Yes
Privacy preserving	No	No	No	No	Yes
Support for sampling	Yes	Yes	Yes	Yes	Yes
Probability of detection	[1-$(1-p)^c$]	[1-$(1-p)^{c*s}$]	[1-$(1-p)^c$]	[1-$(1-p)^c$]	[1-$(1-p)^{c*s}$]

1. n is the block number, c is the sampling block number and s is the numbers of sectors in blocks. p is the probability of block corruption in a cloud server and P_k is the probability of k^{th} cloud server in a multi-cloud.

multiple TPAs to cross check and cross authenticate the data integrity, privacy, and accuracy. There is a trade-off between data dynamics, privacy preservation, and public verifiability in these schemes. TPA may simultaneously handle various audit sessions from different users for their outsourced data files by multi-user setting during the auditing process. Some other notable contributions for DIV which use TPA are discussed as follows:

Rizvi et al. [51] proposed a TPA based integrity checking solution which uses time-released session keys and SLA for each auditing service of CSP. Their simulation results show a heavy decrement in performance for some malicious attempts from the TPA. Another

Table 2.4: Comparison of Different Data Auditing Techniques with TPA

Properties	**Wang et al [18]**	**Wang et al [19]**	**Hao et al [20]**	**Co-PDP [50]**
Primitives	Bilinear Map, MAC, Homomorphic Authenticator	Merkle Hash Tree, Aggregate Signature	RSA based Bilinear Homomorphic Verifiable Tags	Homomorphic Verifiable Hash Index Hierarchy
Type of Guarantee	Probabilistic	Probabilistic	Deterministic	Probabilistic
Public Verifiability	Yes	Yes	Yes	Yes
Use of TPA	Yes	Yes	Yes	Yes
Data Dynamics	Yes	Yes	Yes	Yes
Privacy Preserving	Yes	Yes	Yes	Yes
Support for Sampling	Yes	Yes	No	Yes
Probability of Detection	[1-$(1-p)^c$]	[1-$(1-p)^{c*s}$]	[1-$(1-p)^{c*s}$]	Z^*

1. n is the block number, c is the sampling block number and s is the numbers of sectors in blocks. p is the probability of block corruption in a cloud server and P_k is the probability of k^{th} cloud server in a multi-cloud.
2. $Z^* = [1 - \prod p_k \epsilon p (1-p_k)^{r_k * c * s}]$

solution is presented by Mei et al. [52] as the Trust Enhanced Third Party Auditor (TETPA). This architecture enables trustworthy auditing which uses different methods like USBKey and Third Party Module (TPM) for security improvement. Huang et al. [53] applied bilinear pairing for the integrity verification, which is audited by the TPA. Their assumption for the solution is that TPA may be malicious in some critical situations. Further, Zhang et al. [54] proposed a Secure Certificate Less Public Verification (SCLPV) scheme for cloud storage, which is more secure against arbitrary adversaries and malicious TPA's. In this technique, TPA does not manage certificates.

In another work, Li et al. [55] presented two new public auditing protocols for low-performance devices in cloud computing. This approach is based on online/offline signatures for lightweight computing. Zhang et al. [56] found two security flaws in public proof of cloud storage based on lattice assumption. It also counterparts the malicious CSP and curious TPA from lattice assumptions. Xiang et al. [57] presented an efficient and dynamic verification approach which works without a TPA. It is a probabilistic approach with limited verification

queries. This scheme also supports the data dynamics with controllable Paillier Encryption. While Yu et al. [58] proposed a key resilience approach for the outsourced key update. This update is performed by TPA, and it is transparent to the cloud users. The validity of key depends on the user and TPA can only see the encrypted version of the secret key.

2.3 DDoS Attack Prevention

In this section, a surveyed of different types of DDoS attack techniques and their drawbacks are presented. Thereafter, various DDoS attack tools and DDoS defense mechanism are discussed for review.

2.3.1 Types of DDoS Attack

There are different types of DDoS attacks which affect the availability and functionality of the Cloud services. Broadly, DDoS attacks can be categorized into two classes [24]: (1) Bandwidth Depletion Attack and (2) Resource Depletion Attack. Description of these attacks are given below.

1. **Bandwidth Depletion Attack:** This attack is also known as "Brute Force Attack [59]." In this type of attack, a significant number of malicious and spurious user's requests consume all the Internet bandwidth that aggregates and overwhelms the entire Internet traffic. It produces a massive distributed attack [60], and affects thousands of nodes. Further, it can be classified in two subcategories such as flood attack and amplification attack. Zombie is an Internet connected computer that has been compromised by the intruder and performs the malicious activity on the victim side with the help of remote connectivity. The zombies initiate flood attack which send a significant amount of traffic to the victim system and simply block the legal services. It depends on User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) packets. In the amplification attack, Zombies or attackers send many messages to a broadcast IP address. It results in a large number of reply messages at the end of victim system because in the subnet, each address includes a response to the broadcast message.

2. **Resource Depletion Attack:** This attack is also known as “Semantic or Vulnerability Attack" [59]. In this type of attack, a malicious attacker creates and exploits the vulnerability in the resource and generates resource starvation conditions. With this condition [60], legitimate user is not able to access the required resources. This type of attack uses Hyper Text Transfer Protocol (HTTP), Hyper Text Transfer Protocol Secure (HTTPS), or Domain Name System (DNS) to exploit the vulnerability of Cloud.

2.3.2 DDoS Attack Tools

There are several DDoS attack tools exists in the literature, which are used to implement DDoS attacks on different systems. Knowledge of attack-characteristics of a tool is required to monitor and detect the DDoS attack. These tools use UDP, ICMP, and Transmission Control Protocol (TCP) to detect and protect the DDoS attack. Some of these tools use channel encryption to provide security. Thus, detection of such attacks is very difficult.

Trinoo [61] is one of the most common UNIX based tool which helps attackers to understand the possible structure, functionalities of defense and anatomy of the DDoS attack. One problem with this tool is that attacker cannot spoof its source address. Another tool, Tribe Flood Network (TFN) [62] is a collection of the various computer programs to perform the DDoS attack on victim side. These programs are written by the security experts and hackers. Further, Stacheldraht [63] is a malware that is written for Solaris and Linux systems. Because of this malware, each affected system can act as DDoS agent.

Mstream [64] is a three-tiered DDoS tool which overburdens the CPU and restricts the network bandwidth. This tool is based on UNIX operating system and exploits many web services by utilizing TCP and UDP vulnerabilities. Further, Shaft [65] tool is a binary tool which works as an agent. If a system is affected with the Shaft then, its distributed nature adds “Many to One Relationship” and many nodes are altered, modified, and/or lost their identities.

Another tool Trinity [66] is a binary agent system which is installed on a LINUX system. It alerts about the list of connected port servers and helps attacker to drop any kind of web service. Knight [67] and Kaitan [67] are Internet Relay Chat (IRC) based tools which are used for UDP flood attack and SYN attacks. They are used as an urgent pointer flooder and

Table 2.5: Comparative Analysis of Different DDoS Attack Tools

Tool Name	**Possible Attacks**	**Packet Format Used to Launch Attack**	**Channel Encryption**
Trinoo [61]	Bandwidth Depletion, Remote Buffer Overrun Exploitation	UDP	Yes
Tribe Flood Network (TFN) [62]	Both Bandwidth and Resource Depletion	UDP, TCP SYN, ICMP echo request and ICMP directed broadcast	No
TFN2K [65]	Bandwidth Depletion, Resource Depletion and Mix Attacks	UDP, TCP SYN and ICMP	Yes (Key Based CAST-256 Algorithm)
Stacheldraht [63]	Both Bandwidth and Resource Depletion	UDP, TCP SYN, ICMP echo request and ICMP directed broadcast	Yes (Symmetric Key)
Mstream [64]	Bandwidth Depletion	TCP RST, ICMP and TCP with ACK flag set	No
Shaft [65]	Both Bandwidth and Resource Depletion	UDP, TCP, ICMP	No
Trinity [66]	Both Bandwidth and Resource Depletion	UDP, TCP SYN, TCP Null TCP RST and TCP with ACK flag set	No
Knight [67]	Both Bandwidth and Resource Depletion	UDP, SYN and pointer flooder	No
Kaitan [67]	Both Bandwidth and Resource Depletion	UDP, TCP SYN and TCP PUSH + ACK	No
BlackEnergy [68]	Both Bandwidth and Resource Depletion	HTTP, HTTPS, XML	Yes
Low-Orbit Ion Cannon (LOIC) [69]	Both Bandwidth and Resource Depletion	HTTP, HTTPS and XML	Yes
Aldi Botnet [69]	Both Bandwidth and Resource Depletion	HTTP, HTTPS and XML	Yes

perform the massive DDoS attack on victim side.

BlackEnergy [68] is a web-based attack tool. This tool can launch different types of DDoS attack and controls the bots using minimum syntax and structure. It is developed by a Russian hacker, who steal very confidential data from secret agency of Russia. Furthermore, Low-Orbit Ion Cannon (LOIC) [69] is open source and network stress testing tool whose application on the network may cause a DDoS attack. It is initially developed by Praetox Technologies and written in C# language.

Finally, Aldi Botnet [69] is a popular botnet ransomware that helps professional hackers to perform DDoS attack and steal confidential information from secret agencies of Switzerland, Germany, and Austria in 2012. Table 2.5 summarizes different DDoS attack tools.

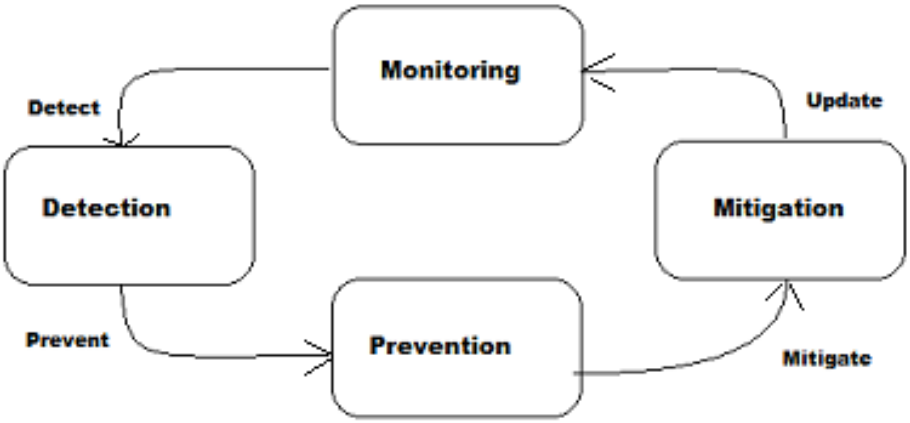


Figure 2.1: Life Cycle of DDoS Defence System

2.3.3 DDoS Defense Mechanism

The DDoS defense mechanism can be divided into four stages; monitoring, detection, prevention, and mitigation. In monitoring, a defender uses various tools to gather network information for execution of protection. The detection step analyzes the executed system for obtaining the source of malicious traffic which causes the perception of attack.

The prevention suggests secure services and intercepts the malicious causes that create critical situation of attack. Finally, the mitigation process evaluates the ultimate effect of the attack and determines the correct response system to manage the DDoS attack effectively. These four steps complete the life cycle of the DDoS defense system as shown in Figure 2.1.

In DDoS defense mechanism, the DDoS attacks are prevented and mitigated in cloud environment. The rate limiting or filtering techniques [70] are used to implement DDoS defense, which can be reconfigured as collaborative and non-collaborative approaches.

Rate Limiting Methods [70] are very convenient to deploy and provision for network nodes. These methods set a dynamic and flexible threshold for filtering the malicious traffic. They are very useful in the detection of many false positives. The primary disadvantage of this method is that it requires coarse grain filtering, which also eliminates the legitimate traffic. Because of this, the detector can not filter the malicious traffic completely.

Filtering techniques are very flexible in adding and removing the IP addresses and convenient for provisioning the network nodes. The main difficulty with the filtering techniques is that they are not able to distinguish between legitimate and malicious traffic generated from

the same source. Another problem with filtering techniques is that they are not able to detect all malicious addresses.

The non-collaborative approaches [70] use reconfiguration for elastic and efficient DDoS defense. These approaches are distributed in nature and provide reliable solution because users can create different instances of Virtual Machines (VMs) in various data centers. The major difficulty with this approach is to put defense mechanism in the right place for defense. These techniques require precise information and orchestration about the network topology and resources for the selection of right place.

On the other hand, the collaborative approaches use push-back and cooperative firewalls which identify and reach to the closest source of the attack. It appraises robust defense and mitigation architecture. The main difficulty with this approach is collaboration and trust establishment between different domains. Another difficulty is its inability to automatically adjust defense mechanism based on the complexity and variety of DDoS attacks.

Rajan et al. [10] proposed an approach that favours authorized traffic over the malicious traffic. This scheme uses suspicious assignment and scheduler for DDoS prevention. The suspect traffic diagnosis of DDoS attack with this approach requires around 0.1 second to 10 seconds.

Further, Chu et al. [71] presented OpenFlow-based DDoS defender for automatic self-defense. In this scheme, two static thresholds are used. When these thresholds met, it indicates the DDoS attack. The diagnosis time of suspicious traffic related to the DDoS attack for this approach is around 0.8 second to 14 seconds.

On the other hand, Choi et al. [22] proposed the Content-Oriented Networking Architecture (CONA) that examines the requested content and initiates the countermeasure for DDoS attack. The diagnosis time of suspicious traffic related to the DDoS attack for this approach is around 0.6 second to 16 seconds.

In another work, Braga et al. [21] proposed a lightweight traffic flow feature that uses Self-Organizing Maps (SOM) based neural network for flow prediction. This network uses an OpenFlow NOX controller for transformation and analysis of classifier modules. This module is efficient, scalable, and updated to address new vulnerabilities.

Furthermore, Lua et al. [23] presented an intelligent, fast flux swarm network that reorganizes the system and provides maximum availability to all clients and servers. In this

Table 2.6: Comparative Analysis of Different DDoS Defense Schemes

Techniques	Mitigation Policy	Protection at Source End	Protection at Core	Protection at Victim End	Strategy Model	Mitigation Scheme
Chu et al. [71]	Non-Collaborative	No	Yes	No	Static	Rate Limiting
Braga et al. [21]	Non-Collaborative	No	Yes	No	Static	Filtering
Choi et al. [22]	Non-Collaborative	No	Yes	No	Static	Rate Limiting
Lua et al. [23]	Non-Collaborative	No	No	Yes	Network Reconfiguration	Filtering
Yao et al. [72]	Non-Collaborative	Yes	No	No	Static	Filtering
Dou et al. [73]	Non-Collaborative	No	No	Yes	Static	Filtering
Shin et al. [74]	Non-Collaborative	No	Yes	Yes	Defense Reconfiguration	Filtering
Zarger et al. [75]	Collaborative	Yes	Yes	No	Cooperative Firewalls	Rate Limiting
Lee et al. [59]	Collaborative	Yes	No	No	Push-back	Rate Limiting
Yu et al. [76]	Non-Collaborative	No	No	No	Hybrid Reconfiguration	Filtering
Saxena et al. [77]	Collaborative	Yes	No	Yes	Cooperative Firewalls	Rate Limiting

scheme, a parallel optimization algorithm runs for constant reconfiguration of the swarm network, which provides efficient DDoS detection in the Cloud environment.

Zarger et al. [75] demonstrated an approach for distributed, and collaborative defense against the DDoS attack. This method detects the high volume of malicious traffic during the DDoS attack. It also detects the closer source of the attack. The diagnosis time of suspicious traffic related to the DDoS attack for this approach is around 0.7 second to 20 seconds.

In another work, Yao et al. [72] suggested Virtual Source Address Validation Edge (VAVE) that expands the Software Defined Networking NOX (SDN-NOX) controller on

the virtual machines for flow check entries, filtering, and validation for DDoS prevention. The VAVE modules validate the incoming packet and take decisions.

Dou et al. [73] suggested a Confidence-Based Filtering (CBF) technique for Cloud computing. It produces a simple profile in the normal (non-attack) period. In this approach, each packet is examined on the basis of some correlation parameters.

Shin et al. [74] presented a framework "FRESCO" that produces a prototype of the application to help with detection and mitigation of security modules. The design of "FRESCO" is built on OpenFlow switch security modules. This security module merges the services and gives an effective defense for complex networks. FRESCO provides an Application Programming Interface (API) to develop, access, and generate statistics of network flow.

Further, Lee et al. [59] suggested a collaborative defense model called as "CoDef" for prevention of DDoS attack. Collaborative routing and rate control are the two underlying mechanisms of this approach. This method is based on special route controller for the participation of every Application Server (AS). The AS helps in the categorization of high priority flow, low-priority flow, and filtered flow, which plays a measurable role in the prevention of DDoS attack.

Yu et al. [76] described a defense reconfiguration mitigation technique which works with multiple parallel Individual Pocketed Spring (IPS) system. This method adds new features to service reconfiguration mitigation approach and provides the maximum availability for all clients and servers. It identifies malicious traffic efficiently and decreases the number of service VMs and IPSs when malicious DDoS traffic are reduced.

Table 2.6 shows the comparative analysis of the DDoS defense methods. Other notable contributions for DDoS mitigation are discussed as follows.

Kalkan et al. [78] presented a collaborative and proactive DDoS defense method. It is a statistically based protection method. The selection of attributes during the current attack traffic is the distinctive property of this approach.

Further, Wang [79] suggested an elastic and resilience defense mechanism to prevent DDoS attacks. This work protects the Domain Name System (DNS) authoritative name servers with signal domain redirection.

Singh et al. [80] reviewed different Internet Protocol (IP) packet trace-back schemes for DOS attacks prevention with their advantages and disadvantages. It inferred that in over-

all DDoS mitigation process IP trace-back approach is important but less used. It is also suggested that skillful integration of different IP trace-back techniques is a good research prospect for DDoS attack mitigation.

In another work, Singh et al. [81] worked on application layer HTTP-GET Flood DDoS attack. The primary mapping results are showing the weakness and limitations of these types of attacks and also motivates to deal with attack enthusiastically.

Further, Nunes et al. [82] suggested us a Belief-Desire-Intension (BDF) architecture based solution of DDoS attack. This architecture separates the deliberative state of the system from its motivational state for DDoS detection. Somani et al. [83] proposed an auto-scaling algorithm for resource allocation in the situation of DDoS attack. Under DDoS attack, the resources are "overloaded". This approach suggests a way for horizontal scaling, vertical scaling, and their migration. They also addressed issues like multi-tenancy, migration, resource race, performance isolation, and interference for VM's used in cloud services.

Wang et al. [84] presented an architecture called as DDoS attack mitigation architecture using software defined networking (abbreviated as DaMask). This architecture requires little effort to change current cloud computing service architecture for better security. It provides a graphical model for anomaly detection, which is used in DDoS prevention. Finally, Sieklik et al. [60] proposed a methodology to set up a Trivial File Transfer Protocol (TFTP) for DDoS amplification attack and its verification. In this work, a variety of countermeasures are discussed to limit the severity of the attacks effectively.

2.4 Summary

This chapter surveyed existing techniques for authentication, data integrity verification, and DDoS attack prevention in cloud environment. From the study of existing authentication techniques, we can note that conventional authentication methods are not best suited for cloud environment. Further, some existing methods which are specifically designed for cloud environment are susceptible with different attacks like Man in the Middle Attack, Impersonation Attack, Reply Attack. Some techniques require huge computational overhead due to their complex hash functions. We have also presented existing PDP and PoR techniques for data integrity verification. The majority of these techniques do not support dynamic data

operations and batch auditing. Further, these techniques are computation intensive, and effectiveness of these approaches rely on the preprocessing of the data. Some TPA based data integrity verification techniques are also discussed. Sometimes these techniques suffer from privacy preservation problem. Finally, we have reviewed different types of DDoS attacks and their prevention and mitigation mechanisms in cloud environment. The main difficulty with the majority of non-collaborative approaches is that they require precise information and orchestration about network topology and resources. On the other hand, defence mechanism in collaborative approaches rely on the trust establishment between different domains. Hence, we need to propose efficient methods for authentication, data integrity verification, and DDoS attack prevention.

Chapter 3

Efficient Authentication

Authentication is the truth confirmation of an attribute or data, which is claimed true by an entity. This part of the research work describes effective authentication mechanism in cloud environment. In our approach, user's data is stored as file blocks into different cloud servers. To provide better security, we are going to propose an authentication mechanism based on bilinear pairing. A Secret Key Generator (SKG) is used to maintain a set of identities for users and to perform authentication in cloud. This set of identities helps to form identity hierarchy and generate the secret keys for different users. The proposed mechanism also prevents malicious users to access legitimate resources. The detailed description of the proposed approach is given in this chapter.

In section 3.1, notations and preliminaries related to our work are described. Section 3.2 discusses the proposed work for efficient authentication in cloud environment. We have analyzed different types of attack with test cases in section 3.3. Finally, section 3.4 summarizes the chapter.

3.1 Notations and Preliminaries

In this section, first, we introduce the necessary notations. Then, we briefly describe the preliminaries of bilinear pairing method which is used for secret key generation in this work.

3.1.1 Basic Notations

The definition of the basic notations help readers to understand and follow the proposed approach in subsequent sections. The definition of the frequently used notations are given in the following.

1. h : The maximum height of the hierarchy for authentication system, which is maintained by Secret Key Generator (SKG).
2. id : Identity tuple (id_1, $id_2 \ldots id_j, \ldots id_n$) created from user's identities and is maintained by SKG.
3. P : A set of public parameters which are used to communicate with different cloud service providers.
4. k : Security parameter which is used to generate local parameters and MSK.
5. PK : Public Key.
6. SK_{id_j} : secret key for the user j.
7. $E_{SK_{id_j}}(\bullet)$ and $D_{SK_{id_j}}(\bullet)$: Encryption and decryption functions used for the user j with SK_{id_j} .
8. F_{j_k} : The k^{th} file of user j.
9. C_{j_k} : The k^{th} Cipher file of user j.
10. G_1 and G_T : Cyclic and multiplicative groups of prime order p.
11. G_2 : Multiplicative group of prime order p, which may be cyclic or acyclic.
12. $BP(\cdot, \cdot)$: Bilinear pairing function which is used to generate secret keys and public parameters for cloud users.

3.1.2 Bilinear Pairing

Bilinear pairing maps two cryptographical groups into a third group for construction or analysis of cryptographic systems. We use the bilinear pairing in our work to generate secret keys and public parameters for users to communicate with CSPs. These secret keys and public parameters are hard to be re-generated by an attacker due to its computational complexity. In our work, we use asymmetric type of bilinear pairing.

Our assumptions for bilinear pairing [85] are as follows: Let, r be a prime number, and G_1 and G_T are cyclic groups of order r. In this work, we use multiplicative group notations

for this purpose. Let G_2 be a multiplicative group, which is not necessarily cyclic and each element in G_2 has order for dividing r. Thus, bilinear pairing (BP) is a function, which can be computed efficiently using equation 3.1.

$$BP : G_1 \times G_2 \longrightarrow G_T \tag{3.1}$$

The properties of the bilinear function are as follows:

1. **Non-degeneracy:** Let, 1_{G_k} defines first order element in group G_k. Then, with this property, we found that $BP(g_1, g_2) = 1_{G_T}$ for all $g_2 \in G_2$ if and only if $g_1 = 1_{G_1}$. Similarly, $BP(g_1, g_2) = 1_{G_T}$ for all $g_1 \in G_1$ if and only if $g_2 = 1_{G_2}$.
2. **Bilinearity:** For all $g_1 \in G_1$, $g_2 \in G_2$, and $a, b \in \mathbb{Z}$: $BP(g_1^a, g_2^b) = BP(g_1, g_2)^{ab}$. Here, $\mathbb{Z}$ represents a group of prime numbers.

3.2 Proposed Technique

In this section, first, the system model is presented for the hierarchical identity based user authentication. Then, we describe the workflow of our proposed technique.

3.2.1 System Model

We propose a novel authentication technique, which is effective for cloud storage. The proposed model has various levels of security depending upon the trust hierarchy. Our approach verifies and guarantees that the CSP could not learn about any file content stored in the cloud server during the execution of authentication mechanism.

To generate secret key, we require careful selection of the identities from the different users. Hence, the proposed authentication scheme delegates the secret key generation to the secret Key Generator (SKG). The SKG is responsible for the selection of identities and secure transmission of the intermediate secret keys to its lower levels.

SKG also generates the secret keys for users from their public keys (PK). Cloud users can use any identity as their public key, for example: Social Security Number(SSN), AADHAR Number, Email-id, etc. Cloud user is allowed to perform any communication with CSP after authentication using public parameters and encrypting their file with his secret key.

In our scheme, users are synchronized and store their file within multiple CSPs. Public parameters are used for this purpose. In case of data loss from any CSP, this synchronization helps users to retrieve their file.

In the proposed authentication scheme, identities are represented as tuple in SKG. SKG contains a master key MSK and a set of local parameters $L = \{l_1, l_2, l_3.....l_m\}$ for different CSPs, which are dependent on file replication factor m. When a user wants to store a file on different CSP servers, user needs to generate his/her own public parameter set P for these servers. Thus, cloud user j utilizes bilinear pairing of his/her secret key SK_{id_j} and CSP local parameters to compute a set of public parameters $P = \{p_1, p_2, p_3, ...p_m\}$ as described in equations 3.2, 3.3, and 3.4.

$$p_1 = BP(l_1, SK_{id_j}) \tag{3.2}$$

$$p_2 = BP(l_2, SK_{id_j}) \tag{3.3}$$

$$p_m = BP(l_m, SK_{id_j}) \tag{3.4}$$

In equations 3.2, 3.3, and 3.4, BP represents the bilinear pairing. The public parameters are utilized for CSPs registration, file storage, and file retrieval from the different CSPs. After file retrieval, file validation is performed, which is based on $CRC - 32$ algorithm. If it is validated, then file will be forwarded to the user.

3.2.2 Workflow of the Proposed Scheme

The workflow of our proposed authentication scheme in different CSP servers is shown in Fig. 3.1. The proposed method includes different steps which are discussed below.

1. **Cloud User Registration:**

   To interact with different CSPs, first, cloud user needs to register with the SKG. For

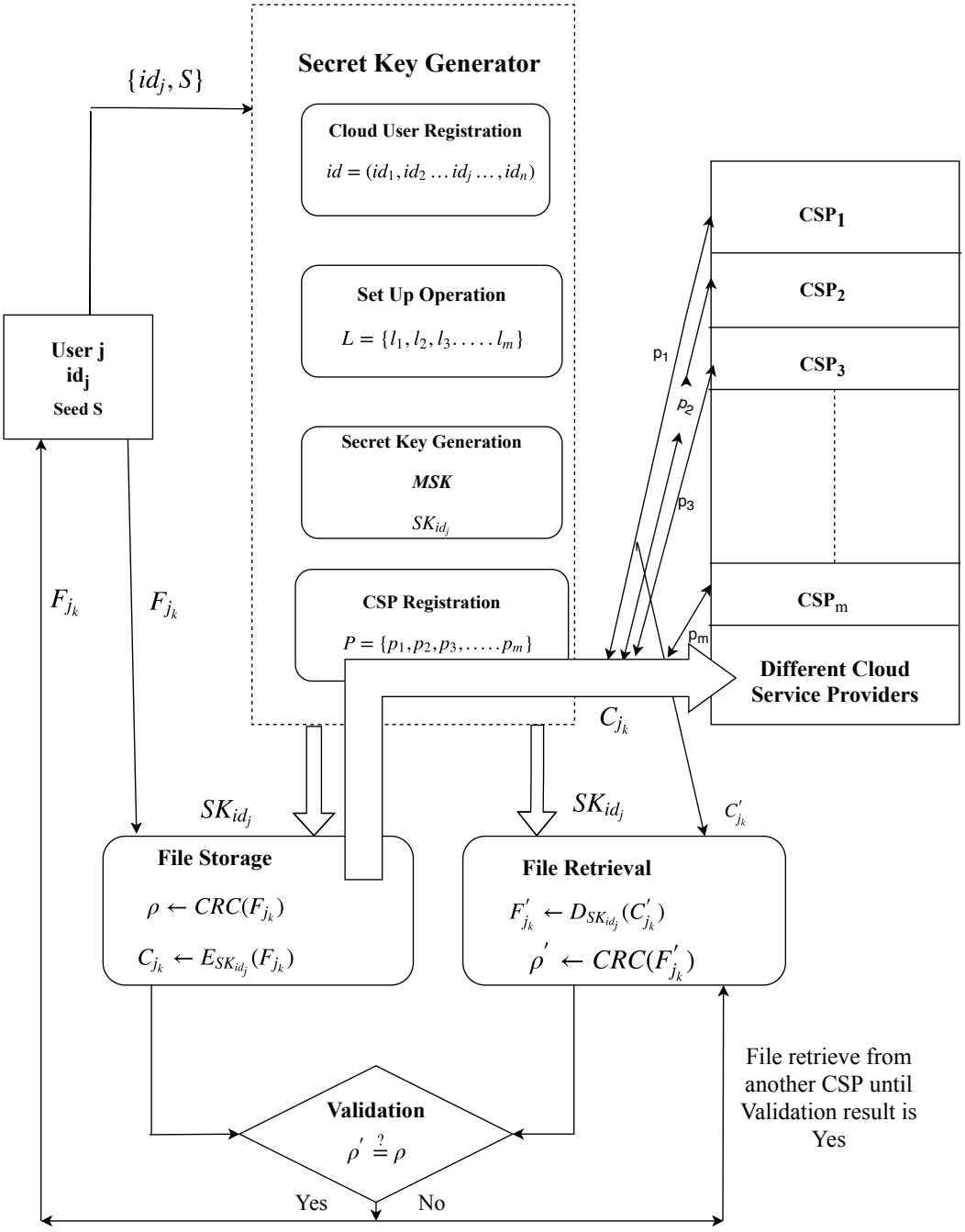


Figure 3.1: Workflow of the Proposed Method

this purpose, the cloud user sends the unique identity id_j and a seed value S to SKG. SKG maintains an identity tuple $id = (id_1, id_2, ..id_j..., id_n)$ for all n registered users. Whenever a new user comes for registration, SKG will check whether the given identity id_j is present in id tuple or not. If id_j is not present in the id tuple, SKG will

successfully register the user by including id_j into the id tuple. If the given identity id_j is already present in the id tuple, SKG will ask user to provide different identity. Next, the seed value will be used for secret key generation.

At the time of registration, user can also select his criteria for identity verification in case of forgotten user identity and seed value. This criteria may be an email alert on his email-id or Short Message Service (SMS) alert on his mobile number.

2. **Setup Operation:**

   The SKG initiates the set-up operation to generate the initial security parameters and local parameters set L for different CSPs. SKG uses a string of 1 or 0 of length k as input and derives the local parameters set L and MSK by randomizing the input. The generated master key and security parameters are only known to SKG. Algorithm 3.1 presents the steps of setup operation.

Algorithm 3.1 Setup Operation for Authentication

Input: $\{0,1\}^k$.
Output: Local parameters set L, Master Secret Key (MSK).

1: Initial Master Secret Key (MSK) generates by $MSK \xleftarrow{R} \{0,1\}^k$.
2: SKG generates a set of local Parameters L by following steps, where $L = \{l_1, l_2, ...l_m\}$.
3: $l_1 \xleftarrow{R} \{0,1\}^k$.
4: $l_2 \xleftarrow{R} \{0,1\}^k$.
5: $l_m \xleftarrow{R} \{0,1\}^k$.

3. **Secret Key Generation:**

   This operation follows h hierarchy levels to generate secret key SK_{id_j} of j^{th} user. In this approach, Secret Key Generator (SKG) module uses a Master Secret Key (MSK) and an identity tuple id = $\{id_1, id_2, id_3.id_j, ..., id_n\}$ of n registered users to generate secret keys. Besides, the j^{th} user provides a seed value S and unique identity id_j to SKG, which are also used to generate his/her secret key through h hierarchy levels. SKG creates a Map Table of height h corresponding to the number of hierarchy levels. Map Table stores h key values which are corresponding to the index of id tuple. These key values are generated randomly based on the seed value S.

   Next, each level generates an Intermediate Secret Key ($IMSK$) using bilinear mapping between the user id and $IMSK$ of previous level except the first level. At the

first level, $IMSK_1$ is generated by bilinear pairing of user identity and MSK. For a particular hierarchy level, user identity is selected from id tuple corresponding to the key value (index of id tuple) stored at an index of the Map Table. This process continues till h^{th} level. At the h^{th} level, $IMSK_h$ is assigned as the final secret key SK_{id_j} for user j.

Figure 3.2 shows the secret key generation process. In this figure, it is mentioned that a Map Table is generated at run time by SKG. According to this Map Table, id_2 is selected from id tuple to generate $IMSK_1$ as the first index stores the key value "2". Thus, at the first level, bilinear pairing of id_2 and MSK generates $IMSK_1$.

Similarly, on the second level, id_5 is selected for generation of $IMSK_2$ from id tuple as the second index of Map Table stores the key value "5". Thus, at the second level, Bilinear Pairing of id_5 and $IMSK_1$ generates $IMSK_2$. This process continues till the h^{th} level. At h^{th} level, id_{1022} is selected from id tuple to generate $IMSK_h$, and bilinear pairing of id_{1022} and $IMSK_{h-1}$ generates $IMSK_h$. Finally, this $IMSK_h$ is set as the secret key SK_{id_j} of user j. Secret key generation algorithm is given in Algorithm 3.2. In this algorithm, an array is used to implement the map table.

Algorithm 3.2 Secret Key Generation

Input: MSK, identity tuple id = (id_1, id_2, id_j, id_n), where $n \geq 1$.
Output: SK_{id_j} is the secret key for the j^{th} user at h^{th} level of hierarchy.

1: **for** $int\ i = 1;\ i \leqslant h;\ i++$ **do** ▷ Map Table Generation
2: $M[i] = Random(S + i)$ ▷ $M[]$ is the Map Table
3: **end for**
4: **for** $int\ i = 1;\ i \leqslant h;\ i++$ **do** ▷ Secret Key Generation
5: **if** i=1 **then**
6: $IMSK_i \leftarrow BP(MSK, id_{M[i]})$.
7: **else**
8: $IMSK_i \leftarrow BP(IMSK_{i-1}, id_{M[i]})$.
9: **end if**
10: **end forreturn** $SK_{id_j} \leftarrow IMSK_h$.

4. **CSP Registration:**

To register with different CSPs, SKG will generate a public parameters set P for these CSPs. Public parameters set P is represented as $P = \{p_1, p_2, p_3.....p_m\}$. In this set, the number of public parameters depend upon the file replication factor m.

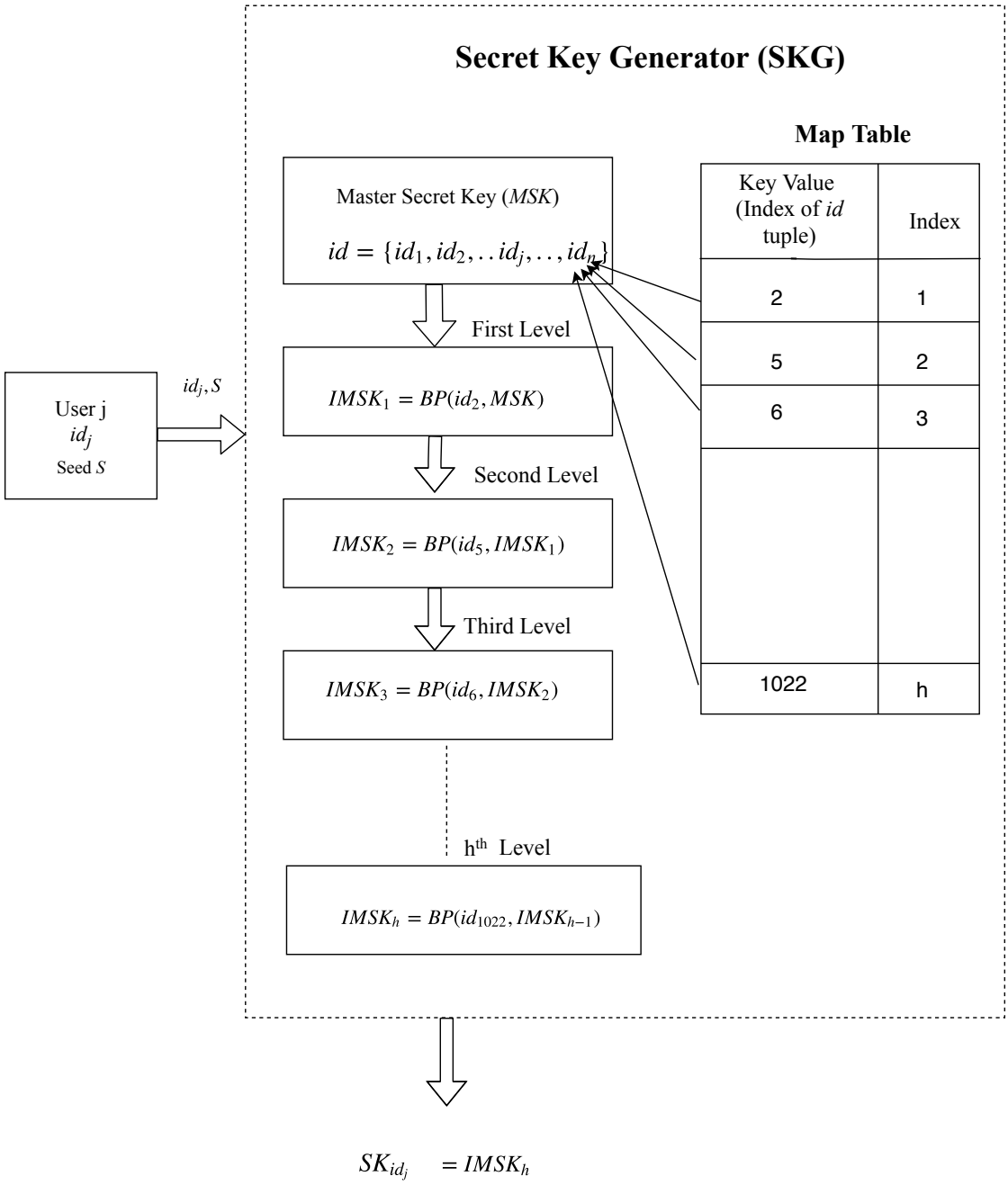


Figure 3.2: Secret Key Generation Process

For the public parameter generation, SKG utilizes the local parameters, which are produced in Set-up operation. For this purpose, SKG induces the public parameter p_m by bilinear pairing of local parameter l_m and his/her secret key SK_{id_j} as shown in equation 3.5. In this process, the public parameter p_m and local parameter l_m are

referred to the m^{th} CSP.

$$p_m \leftarrow BP(l_m, SK_{id_j}) \tag{3.5}$$

5. **File Storage:**

   To store a file into different CSP, user needs public parameter set P and his/her secret key. For this purpose, user follows the secret key generation and CSP registration process. Therefore, user j extracts Cyclic Redundancy Check - 32 (CRC-32) value of the particular file F_{j_k} for future. He/she also stores this CRC-32 value for the file validation. If CRC-32 denotes the output of function $CRC(\bullet)$, then equation 3.6 represents the CRC of file F_{j_k}:

   $$\rho \leftarrow CRC(F_{j_k}) \tag{3.6}$$

   Now, the registered user j encrypts his k^{th} file F_{j_k} with his secret key SK_{id_j}. The output of this encryption is a cipher-file C_{j_k}. The encryption operation $E(\bullet)$ is specified in equation 3.7.

   $$C_{j_k} \leftarrow E_{SK_{id_j}}(F_{j_k}) \tag{3.7}$$

   Here, we consider a multi-cloud environment, in which one file F_{j_k} can be stored into various CSP in synchronized manner to maintain replication factor m. It means the same file F_{j_k} is stored on m different CSPs. Before storing the data into different CSPs, user j has to extract the public parameters set P from CSP registration process for all CSPs. Finally, the encrypted file is stored into different CSPs with the help of these public parameters.

6. **File Retrieval:**

   When cloud user j wants to retrieve his stored file F_{j_k}, he/she must be authenticated with CSP by its public parameter. Thus, user j sends a file retrieval request to a selected CSP with its public parameter p_j. Thereafter, CSP acknowledges to the cloud user with cipher-file C_{j_k}.

User j decrypts the cipher-file C_{j_k} by his/her secret key SK_{id_j}. If the cipher-file is not valid, this algorithm produces $\perp$, otherwise it return with original file $F_{j_k}^{'}$. In the proposed approach, we use standard $D(\bullet)$ notation for decryption process. File decryption process is shown in equation 3.8.

$$F_{j_k}^{'} \leftarrow D_{SK_{id_j}}(C_{j_k}^{'}) \tag{3.8}$$

After decryption, user j takes the CRC-32 of the received file and compares with the ρ. If both CRC-32 are matched then file integrity is preserved and original file F_{j_k} is returned to the user with the help of Algorithm 3.3. Otherwise, file corruption message will be sent to the user.

In case of file corruption, user again requests for file retrieval from other CSPs. This is the advantage of our approach that if the file received from one CSP is corrupted, then user can retrieve it from other CSPs. Any user can take this advantage because SKG maintains m replication factor among different CSPs.

Algorithm 3.3 File Validation

Input: ρ , $F_{j_k}^{'}$.

Output: Validation Result $\rho^{'} \stackrel{?}{=} \rho$.

1: $\rho^{'} = CRC(F_{j_k}^{'})$.
2: Validation $\rho^{'} \stackrel{?}{=} \rho$.

3.3 Security Analysis

In this section, we describe different case studies for security analysis against different types of attack for the proposed method. At the basic level, the security of authentication scheme is formalized such that the adversary should not be able to retrieve original file F from the cipher file C. Here, we assume that an adversary is able to decrypt C, if he successfully retrieves the secret key of user. Here, we have defined different cases corresponding to different scenarios to analyze the possibilities of retrieving the secret key.

- **Case 1: Adversary $\mathcal{A}$ knows the identity tuple (id) and the height of hierarchy**

level (h)

In the first case, adversary $\mathcal{A}$ knows the identity tuple id and height of hierarchy level h. Thus, adversary $\mathcal{A}$ may select a target user t with his/her identity id_t to challenge the security and try to get access by authenticating on behalf of the user. For this purpose, adversary tries to communicate with SKG and CSP with id_t which can be done in different ways to retrieve the SK_{id_t}.

First, adversary $\mathcal{A}$ can put Secret Key Extraction Query (SK_{id_t}) of target t for prediction. After the query generation from adversary $\mathcal{A}$, SKG check this identity in its current tuple. This tuple helps SKG to issue the secret keys. Further, if this identity is already present in the identity tuple, SKG has a record of valid secret key for this identity. This validity depends upon issuing and expiring time stamp value for the secret key. Therefore, SKG discards this query raised from adversary as the legitimate user already has valid secret key and there is no need to generate secret key for this identity.

Further, if an adversary tries to recover SK_{id_t} through forget password mechanism. SKG will send an OTP on user's registered mobile number or send a link to his registered email id since, adversary does not have the access to these criteria. Hence, adversary will not be able to retrieve the secret key.

In our approach, secret key is generated by bilinear pairing of id_j and MSK / $IMSK$ in h levels, and the id_j is selected from the Map Table, which contains the combination of h identities of id tuple. Therefore, an adversary can try to generate Map Table by performing permutation of h ids from the id tuple for retrieving the SK_{id_t}. To generate the Map Table, adversary has to perform $\frac{n!}{(n-h)!}$ permutations, where n and h are the total number of identities and height of the hierarchy level, respectively. Further, for each combination in Map Table, adversary has to guess MSK. Hence, adversary will not be able to generate SK_{id_t}.

Moreover, there is another possibility that the adversary decides to choose adaptive approach for predicting identity verification steps. In adaptive approach, a list of query is dependent on the previous queries along with their answers. But, SKG restricts adversary by allowing the user only for few number of prediction queries. Thus, our authentication scheme is said to be (T, K)-secure against adaptive identity attack, if

adversary $\mathcal{A}$ makes at most K key extraction queries for a certain time period T.

- **Case 2: Adversary $\mathcal{A}$ knows targets (SK'_{id_t}, C_k)**

  This query is placed for guessing the original data block from chosen cipher file. For this query, adversary $\mathcal{A}$ select a random secret key SK'_{id_t} for decrypting a chosen cipher file C_k. If this secret key SK'_{id_t} matches with the secret key SK_{id_t}, then this query returns the resulting original data block F_k. Otherwise, it returns $\perp$ (NULL) as the cipher file cannot be decrypted.

  Adversary can also select adaptive approach in identity tuple id for this purpose. In this approach, each query is dependent on the previous queries along with their answers. But, it is very difficult for secret key SK'_{id_t} to match with the secret key SK_{id_t}, because SKG use bilinear pairing of identity and secret key of the previous user (ancestor) to generate secret key of next user in the hierarchy. This makes very difficult to predict the secret key of any target user for adversary, because it requires all identities to be revealed from SKG. Thus, for each user, intermediate secret keys and secret keys are different. On the other hand, to make the prediction very difficult, SKG restricts adversary by allowing a user only for few number of prediction queries. However, it is another fact that for each user, master key MSK is the same, but seed value S is different. Thus, for each user, Map Table and intermediate secret keys are different.

  Thus, our authentication scheme is said to be (T, D)-secure against adaptive chosen cipher file attack if adversary $\mathcal{A}$ can make at most D decryption queries for a certain time period T.

- **Case 3: Adversary knows id tuple, height h, and the Local Parameters**

  In the third case, adversary $\mathcal{A}$ chooses a random local parameter p_t and secret key SK_{id_t} for prediction of public parameters.

  If a single point failure is occurred then adversary knows id tuple, height h, and the local parameters set P. These local parameters and secret key SK_{id_t} are required to guess the public parameters of CSP. It leaves us in the situation of case 1.

  Thus, our authentication scheme is said to be (T, N)-secure against adaptive chosen public parameter attack if adversary $\mathcal{A}$ can make at most N queries for a certain time period T.

To conclude this security test, we can say that the authentication scheme is a secure

approach for obtaining Data as a Service(DaaS) from different CSP's because it is effectively tested against different types of attack.

3.4 Summary

This chapter proposes an hierarchical-authentication mechanism for maintaining the effective authentication in cloud computing. The proposed method is based on the hierarchy of user identities. Multiple users can authenticate with multiple CSPs simultaneously. In this work, the secret key generation task is delegated to the secret key generator (SKG) which is responsible for the selection of identities at different hierarchical levels and the secure transmission of the secret keys. A number of public parameters are maintained for effective authentication in different CSPs. We achieve an average user performance and CSP performance of 91.2% and 93% for the proposed authentication system. Further Security of the proposed method is tested under different conditions and outcome test result analysis do ensure the security of scheme from various type of attacks so that it proves an effective and secure method for authentication in cloud environment in such a manner that the advisory will not be able to forge into system as well as also capable to identify the unauthorized request.

Chapter 4

Data Integrity Verification

In present era, data and applications are moved towards cloud infrastructure and data centers, which run on virtual computing resources in the form of virtual machines. The use of virtualization on large scale brings additional security overhead for tenants of a public cloud service. Hence, data integrity verification (DIV) with high security and minimal overhead is one of the major prerequisite for the expansion and perception of cloud computing. In this chapter, we are going to propose two different techniques for DIV. The first DIV technique uses algebraic signature (AS), combinatorial batch codes (CBC), and homomorphic tag; whereas the second technique uses Paillier homomorphic cryptography (PHC) system and combinatorial batch codes (CBC).

In section 4.1, notations and preliminaries related to our work are described. Section 4.2 provides the detail description of the first DIV method for cloud environment. Description of the second DIV method is given in section 4.3. Section 4.4 furnishes the initial setup details for both DIV methods. Finally, summary of this work is given in section 4.5.

4.1 Notations and Preliminaries

In this section, first we introduce the basic notations, which are frequently used in our proposed DIV methods. These will help readers to understand the technical terms used in this chapter. Next, the preliminaries AS, HT, CBC, PHC, and some important theorems are described for further discussion.

4.1.1 Basic Notations

1. AS_g (•): Algebraic signature function.
2. f(•) A pseudo-random function (PRF) which maps as follow $f : \{0,1\}^k \times \{0,1\}^l \longrightarrow \{0,1\}^l$.
3. σ(•) A pseudo-random permutation (PRP) which maps as follow $\sigma : \{0,1\}^k \times \{0,1.....n\} \longrightarrow \{0,1.....n\}$.
4. $F = F[1], F[2], ...F[i], ...F[n]$: F represents a file, $F[i]$ denotes the i^{th} data block of file F, and n represents the total number of data blocks in file F.
5. L: Length of a bit string.
6. v: Number of verification request.
7. r_1, r_2: Random numbers chosen from the Galois field.
8. g^i: A number selected from a Galois field G corresponding to the i^{th} data block.
9. $T = T[1], T[2]..T[i], ...T[t] : T$ denotes all block tags, $T[i]$ represents the i^{th} tag of T, and n is the number of data blocks.
10. R: Number of blocks chosen for challenge operation.
11. $\mathcal{C}$: Combinatorial batch codes.
12. $\mathcal{X}$: Set of n elements (or items).
13. $\mathcal{S}$: Collection of m subsets of $\mathcal{X}$.
14. H(•): A hash function.
15. ϕ(•): Euler's totient function.
16. $\mathbb{Z}$ and $\mathbb{Z}^*$: Group on integer numbers.
17. r: Number of deleted blocks from total file blocks.
18. x and y: Two different odd prime numbers of the same length.
19. λ : Multiplication of two prime numbers x and y.

4.1.2 Algebraic Signature

Algebraic signature uses symmetric key techniques to enhance the efficiency of security methods. The execution of algebraic signature can achieve high speed from tens to hundreds of megabytes per second. This allows challenger to verify data integrity by comparing only the response returned by the storage server. For this purpose, challenger does not require

the entire original data for verification because algebraic signature uses only small amount of data for challenges and responses. Further, the efficiency of algebraic signature schemes permits the construction of large-scale distributed storage systems in which large amount of data can be verified with maximal efficiency and minimal overhead. The aggregation and algebraic properties of the algebraic signature provide extra benefit for batch auditing in our design.

Algebraic signature [25] of file blocks $F[1], F[2], ..., F[n]$ is a kind of hash functions as defined in equation 4.1

$$AS_g(F[1], F[2], ..., F[n]) = \sum_{i=0}^{n} F[i].g^i \tag{4.1}$$

where, g^i is the number corresponding to the i^{th} data block, which is selected from a Galois field G.

Algebraic signature [86] itself is a single string. The data compression rate of the algebraic signature is $DCR = \frac{F[i]}{L}$, where $F[i]$ is the size of a file block and L is the length of a string. For example, if the size of a file block $F[i]$ is 1 KB and $L = 64$ bits, then corresponding algebraic signature is 64 bits and $DCR = \frac{F[i]}{L}$ = 128. Hence, the data compression rate of algebraic signature is 128.

Algebraic signature satisfies the following property.

Property: The sum of signatures of two or more individual files/file blocks is equal to the signature of the sum of corresponding file/file blocks. If $AS_g(X)$ and $AS_g(Y)$ are the algebraic signatures of two files X and Y, respectively, then

$$AS_g(X + Y) = AS_g(X) + AS_g(Y) \tag{4.2}$$

Proof: The property of algebraic signature can be verified as follows.

In this property, $x_1, x_2, ...x_n$ and $y_1, y_2, ...y_n$ are the file blocks corresponding to the file X and Y, respectively.

$\Rightarrow \quad AS_g(X) + AS_g(Y).$

$\Rightarrow \quad AS_g(x_1, x_2, ...x_n)$ + $AS_g(y_1, y_2, ...y_n).$

$\Rightarrow \quad \sum_{i=1}^{n} x_i.g^i$ + $\sum_{i=1}^{n} y_i.g^i.$

$$\Rightarrow \quad \sum_{i=1}^{n}(x_i + y_i).g^i.$$

$$\Rightarrow \quad AS_g(X + Y)\ .$$

We use this property of algebraic signature for tag generation, which can be calculated solely using the signatures of the file blocks.

4.1.3 Homomorphic Tag (HT)

Homomorphic Tag is used to grant access and authentication of file blocks. HT introduces the "labelled data" for file blocks which is generated by integrating the algebraic signature of different file blocks. Further, HTs are also encrypted with a secret key. HT is verifiable by the person who knows the secrete key. HT also guarantees that TPA will not learn any information about the data content stored in the cloud servers during the auditing process. It also helps to perform dynamic data operations in the cloud environment.

4.1.4 Combinatorial Batch Codes (CBC)

Combinatorial Batch Code $\mathcal{C}$ (n, N, k, m, t) [87] is a set system ($\mathcal{F}$, $\mathcal{S}$), where $\mathcal{F}$ is a set of n file blocks, $\mathcal{S}$ (called servers) is a collection of m subsets of $\mathcal{F}$, and $N = \sum_{s\epsilon\mathcal{S}} |s|$, such that for each k-subset $\{F[i_1], F[i_2], ..., F[i_k]\} \subset \mathcal{F}$ there exists a subset $\mathcal{C}_i \subseteq \mathcal{S}_i$, where $|\mathcal{C}_i| \leqslant t$, i= 1,.....,m, such that

$$\{F[i_1], F[i_2], ..., F[i_k]\} \subset \bigcup_{i=1}^{m} \mathcal{C}_i \tag{4.3}$$

If we fix $t = 1$; it means that CBC permits only one item to retrieve from each server. This CBC is denoted as an (n, N, k, m)-CBC.

In our proposed work, we use CBC as follows: Let, a file F consisting n blocks is to be stored among m servers in such a way that any k file blocks out of the n file blocks can be recovered by retrieving at most t blocks from each server, and the total number of file blocks stored in m servers is N. Our aim is to find out optimal CBCs [88] for which minimal N is required for given values of n, m, k, and t.

We have defined CBC with set systems, and its points are symbolized with file blocks stored in a database and the servers are symbolized by subsets of these points.

For example, (6, 12, 4, 4, 3)-CBC is represented by a matrix $P_{6\times4}$ as shown in equation 4.4, in which 6 blocks of a file are stored among 4 servers in such a way that any 4 file blocks can be recovered by retrieving at most 3 file blocks from each server, and total number of file blocks stored among 4 servers are 12.

In the matrix given in equation 4.4, each row and column correspond to a file block and a server, respectively. Entries in the matrix represent whether a file block is stored in a server or not. Here, the value 1 at the $(i, j)^{th}$ entry denotes that the i^{th} file block is stored into the j^{th} server. The CBC set system can be defined for the above matrix as follows: $\{\{1, 5, 6\}, \{2, 5, 6\}, \{3, 5, 6\}, \{4, 5, 6\}\}$.

$$P_{6\times4} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \tag{4.4}$$

From this representation of set system, we can say that file blocks {1,5,6}, {2,5,6},{3,5,6}, and {4,5,6} are stored in server 1, 2, 3, and 4, respectively. On the other way, we can say that the file blocks 1, 2, 3, and 4 are stored only in server 1, 2, 3, and 4, respectively and file blocks 5 and 6 are stored in all servers.

Need of Dual System:

In the above CBC set representation, if a CSP wants to retrieve a particular file block from the servers, it needs to search all subsets of the servers. This will be very inefficient in the above set system representation. Hence, CSP needs a dual set system representation for the implementation of CBC.

In this dual set system, the servers are symbolized by points, and each file blocks in the database are symbolized by a set containing the points (the servers) which store these items. It is acceptable for the dual set system to contain “repeated blocks". This occurs when two (or more) items are assigned to the identical set of servers. In this representation, CSP can easily find out where a particular block is stored among different servers.

We denote, ($\mathcal{S}$, $\mathcal{F}$) as a dual set system of a set system ($\mathcal{F}$, $\mathcal{S}$), where $\mathcal{F}$ is corresponding to file blocks, and $\mathcal{S}$ is corresponding to servers.

The block of dual set system (n, N, k, m, t)-CBC is expressed by an incidence matrix which can be obtained by transposing the set system matrix. Here, the $(i, j)^{th}$ entry of the incidence matrix represents whether the i^{th} server contains the j^{th} file block or not. The value 1 denotes that the i^{th} server contains the j^{th} file block. The incidence matrix of (6, 12, 4, 4, 3)-CBC is shown in equation 4.5.

$$Q_{4\times 6} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \tag{4.5}$$

As stated above, the CBC set system for the matrix $P_{6\times 4}$ is $\{\{1, 5, 6\}, \{2, 5, 6\}, \{3, 5, 6\}, \{4, 5, 6\}\}$ and the dual set system obtained from the incidence matrix ($Q_{4\times 6}$) of $P_{6\times 4}$ is $\{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2, 3, 4\}, \{1, 2, 3, 4\}\}$. From this dual set system, it can be easily known to the CSP that file blocks 1, 2, 3, and 4 can retrieved only from server 1, 2, 3, and 4, respectively while file blocks 5 and 6 can be retrieved from any server. Thus, it is clear to CSP that 1, 2, 3, and 4 can not be recovered if any single point failure error occurs.

An important fact about the incidence matrix of an (n, N, k, m)-CBC is that it is a $m \times n$ matrix with entries in {0,1}. For each k column, there are k rows such that the resulting $k \times k$ sub-matrix has minimum traversal containing only 1's (i.e., a group of k units in different columns and rows that all accommodate the entry 1). This fact leads to Lemma 4.1.1.

Lemma 4.1.1 *The incidence matrix of an (n, N, k, m)-CBC represents by $m \times n$ matrix of {0, 1} if and only if, for any k columns, there exists a $k \times k$ sub-matrix which has minimum one traversal involving k ones.*

This result is similar to Hall's marriage Theorem [89], which is specified in Theorem 4.1.2

Theorem 4.1.2 *Suppose $(\mathcal{F}, \mathcal{S})$ is a set system. Then, any group of k file blocks*

$\{F[1], F[2], ..., F[k]\} \in \mathcal{F}$ *has a system of distinct representatives* $SDR(i)$*, for all* i*,* $1 \leq i \leq k$*, if and only if the following requirement is satisfied:*

for any group of i *blocks* $F[j_1], F[j_2], ..., F[j_i]$, $|\bigcup_{l=1}^{i} \mathcal{F}_{j_l}| \geq i$.

In the above incidence matrix, it has been easily observed that for $1 \leq i \leq 4$, any i blocks of the dual system include minimum i points. Therefore, we have a CBC with $k = 4$.

4.1.5 Paillier Homomorphic Cryptography (PHC) System

We have also used a variant of PHC system [90] for encryption and decryption in our second DIV method. In PHC system, Group of integer numbers ($\mathbb{Z}_N$ and $\mathbb{Z}_N^*$) are utilized in such a way that $\mathbb{Z}_N \times \mathbb{Z}_N^*$ is isomorphic to $\mathbb{Z}_N^{*\ 2}$. PHC system has three parts which are described in the following paragraphs.

1. Key-Generation

In the first part, PHC system generates public and private keys for encryption and decryption, respectively, and a random number is used to encrypt plain-text. It applies Euler's totient function on two different odd prime numbers to generate these keys. The procedure of key-generation is summarized below.

1. An entity chooses two different odd prime numbers p and q of the same length.
2. Evaluate $J = pq$ and Euler's totient function on J as $\phi(J) = LCM[(p-1)(q-1)]$, where LCM denotes Least Common Multiple.
3. Assure that
   (a) $gcd(J, \phi(J)) = 1$.
   (b) For any integer $a > 0$, we have $(1+J)^a = (1 + aJ) \mod J^2$.
   (c) As a consequence, the order of $(1+J) \in \mathbb{Z}_{J^2}^*$ is J i.e. $(1+J)^J = (1 \mod J^2)$ and $(1+J)^a \neq (1 \mod J^2)$ for any $1 < a < J$.
4. Selects a random number $r \epsilon\ \mathbb{Z}_J^*$ such that $\mu = gcd(L(r^J \mod J^2), J) = 1$, where $L(x) = (x-1)/J$.
5. Returns public key (J, r), private key $(\mu, \phi(J))$, where r is a random number generated from the system.

2. Encryption

The second part of PHC system encrypts plain text using public key. Let $m \in \mathbb{Z}_J$ be

a plain-text to be encrypted and $r \in \mathbb{Z}_J^*$ be a random number. With the definition of isomorphism, cipher-text C can be obtained by function f that maps plain text as:
$\mathbb{Z}_J \times \mathbb{Z}_J^* \longrightarrow \mathbb{Z}_{J^2}^*$ and
$C = E(m\ mod\ J, r\ mod\ J) = f(m,r) = [(1+J)^m.r^J \mod J^2]$; where $C \in \mathbb{Z}_{J^2}^*$.

3. Decryption Algorithm

In the last part of PHC system, user can efficiently decrypt the encrypted text using its private key $(J, \phi(J))$. Decryption steps are given as follows.

1. Set $\hat{C} := [C^{\phi(n)} \mod J^2]$, where C is cipher-text .
2. Set $\hat{M} := (\hat{C} - 1)/J$. (Note that all this is carried out over the integers.)
3. After decryption, plain-text is given by $M := [\hat{M}.\phi(J)^{-1} \mod J^2]$

4.1.6 Some Useful Theorems

In this part, we introduce some existing theorems and lemmas which are required to prove the correctness of the proposed scheme.

Theorem 4.1.3 *Binomial Theorem [91]: If we take $x; y \in \mathbb{R}$, and $n \in \mathbb{N}$, then*

$$(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} (x)^{n-k}.(y)^k \tag{4.6}$$

where $\binom{n}{k} = \frac{n!}{k!.(n-k)!}$

Theorem 4.1.4 *Chinese Remaindering Theorem [92]: If we take some positive integers such that $n_0; n_1; ....; n_{k-1}$. These integers are pairwise co-prime. We also take a different set of integers $x_0; x_1; ....; x_{k-1}$. Then, we find the following congruence:*

$$y \equiv x_i\ (mod\ n_i) \tag{4.7}$$

where $0 \leq i \leq k-1$ and y has a unique solution.

$$y \equiv x_0 \mathbb{N}_0 \mathbb{N}_0^{-1} + ... + x_{k-1} \mathbb{N}_{k-1} \mathbb{N}_{k-1}^{-1}\ (mod\ n) \tag{4.8}$$

Where $n = n_0 \times .... \times n_{k-1} = n_i \mathbb{N}_i$ *and* $\mathbb{N}_i \mathbb{N}_i^{-1} \equiv 1(mod\, n)$ *such that* $0 \leq i \leq k-1$.

Lemma 4.1.5 *Hensel lifting [91]: If we take* y *as a prime number, and* $L(x) \in \mathbb{Z}[X]$ *with an assumption:*

$$L(x) \equiv P_1(X) L_1(X) (mod\, y) \tag{4.9}$$

In this case, $P_1(X)$ *and* $Q_1(X)$ *are the relative prime in* $\mathbb{Z}_y[X]$.

We calculate an integer t *such that* $t \succ 1$. *Then, some polynomials exist such that* $P_t(X); Q_t(X) \in \mathbb{Z}_y[X]$

$$L(X) \equiv P_t(X) L_t(X) (mod\, y^r) \tag{4.10}$$

Where $P_k(X) \equiv P_1(X) (mod\, y)$ *and* $Q_k(X) \equiv Q_1(X) (mod\, y)$

Theorem 4.1.6 *Fermat's little theorem [92]: If we take x as a prime number and another integer* $y \in \mathbb{Z}$ *such that* $gcd(y, \mathrm{x}) = 1$*, then*

$$(y)^{x-1} \equiv 1 (mod\, x) \tag{4.11}$$

4.2 DIV Method 1: Using Algebraic Signature, CBC, and Homomorphic Tag

In this section, first, we describe the system model for DIV method 1. Then, we discuss the DIV technique with algebraic signature, CBC, and homomorphic tag. Thereafter, dynamic data operations and proof of batch auditing are given to show the usefulness of this method.

4.2.1 System Model

We divide the system model into three tiers, which are described in the following paragraphs.

1. **Cloud Users:** Any end user, who wants to use cloud services, may be interpreted as a cloud user. We assume that these cloud users have limited resources. Thus, cloud

user is not capable to perform computation intensive tasks such as data integrity and privacy preservation audits.

2. **Multiple TPA:** TPA is an authorized and authentic entity that is responsible for data integrity verification and privacy preservation. It also takes care for the SLA and legal issue related to data migration. For taking inputs from users, one TPA is assigned the responsibility of TPA moderator which distributes input load to other TPAs. Another TPA is kept as a standby TPA which will act as a TPA moderator in case of single point of failure occured on the moderator. Thus, multiple TPAs are used to achieve load balancing and to perform batch audit in cloud environment.
3. **Cloud Service Provider:** In this group, cloud service providers have established enough infrastructure and resources to provide data storage as a service for cloud customers. These resources may be distributed within many servers which are situated at different locations.

4.2.2 Proposed Technique

We propose a distributed multiple third party data auditing technique. In this technique, multiple TPAs share the load responsibility of single TPA by load balancing. This DIV method works in two phases. In the first phase, file is stored into CSP and in the second phase, file is retrieved from the server and verified for integrity check. Different steps of Phase 1 and Phase 2 of this approach are shown in Figure 4.1.

Phase I: Storing File F at CSP End

In this phase, user assigns a file F to CSP for storing. This file is converted into file blocks before storage. To maintain reliability, CSP also manages homomorphic tags for each file block. It helps CSP to bifurcate file blocks among different servers. After load balancing, CSP returns a set of encrypted HTs to the user. All the above steps are described below.

1. **Request for Data Storage:**

   To store a data file F, cloud user needs to register and authenticate with the CSP. Authentication of user is done by using the method proposed in chapter 3. After authentication, user sends a request to the CSP for data storage of file F.
2. **Setup Operation:**

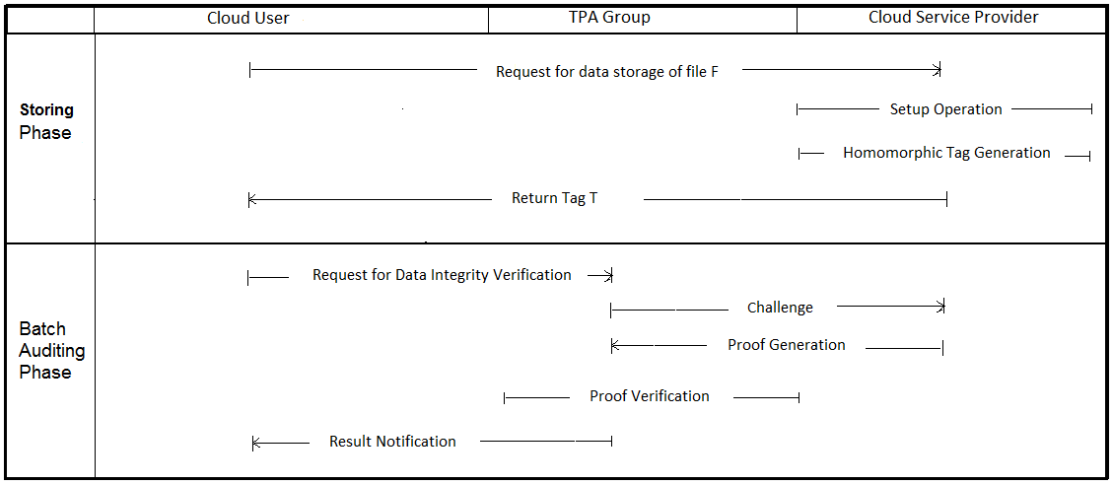


Figure 4.1: Workflow of the Proposed DIV Method 1

Initially, CSP performs the setup operation. This operation initiates the public encryption key of TPA group and generates some random numbers. CSP also converts file F into n file blocks by file blocking operation before storing file blocks among different servers. These steps are summarized in Algorithm 4.1.

Algorithm 4.1 Setup operation

Input: $\{0,1\}^k$.
Output: Public Encryption Key of TPA Group k_t , Random numbers r_1, r_2...r_i .

1: Public Encryption Key of TPA Group k_t generates by $k_t \xleftarrow{R} \{0,1\}^k$.
2: File Blocking Operation
3: $F \rightarrow \{F[1], F[2].....F[n]\}$

3. **Homomorphic Tag Generation:**

   To generate the homomorphic tag, CSP computes algebraic signature corresponding to each file blocks $(F[1], F[2], ...F[n])$. This algebraic signature is treated as the HT for each file block. At any point of time, algebraic signature of a block is not equal with the algebraic signature of another block. Thus, algebraic signature of a block is unique for each block. This property ensures the security of the proposed scheme against insider attacks. Algorithm 4.2 describes the homomorphic tag generation procedure.

4. **Load Balancing and Storing the File F:**

   In this part, first, CSP uses CBC technique for load balancing of all file blocks

Algorithm 4.2 Homomorphic Tag Generation

Input: All file blocks $F[1], F[2], ...F[n]$.
Output: Entry {T} .

1: **for** $int\ i = 1;\ i \leqslant n;\ i++$ **do** ▷ Homomorphic Tag Generation
2: $T[i]$ = $AS_g(F[i])$
3: **end for**
4: T = $\{T[1], T[2], ...T[n]\}$, Where T is the set of Homomorphic Tags.

$(F[1], F[2], ...F[n])$ among m different servers. These file blocks are attached with their homomorphic tags. CSP also encrypts each HT using public encryption key of TPA group k_t, which is derived in set-up operation. This encryption operation for i^{th} tag is given in equation (4.12).

$$\partial_i = E_{k_t}(T[i]) \tag{4.12}$$

Thus, CSP also computes the set of encrypted HTs as ∂ = $\{\partial_1, \partial_2, ...\partial_i.....\partial_n\}$.

As stated earlier in 4.1.4, the load balancing solution of this DIV method is described in the following manner.

Let, a file F of n blocks is to be stored among m servers in such a way that any k of n file blocks can be recovered by retrieving at most t blocks from each server, and the total number of file blocks stored in m servers is N. Thus, CSP can find out an optimal CBC solution to get minimal N value for a given value of n, m, t, k, and replication factor RF. With the help of this solution, CSP stores all file blocks among m servers.

5. **Return Set of Encrypted Homomorphic Tags to User:**

   CSP returns the set of encrypted HTs $\{\partial\}$ to the cloud user. Although these tags increase the size of metadata related to file, yet these tags help cloud user to perform dynamic operations on their data without disclosing the content to anyone. These tags also help TPA group for data integrity verification, which is described in Phase II.

Phase II: Data Integrity Verification

1. **Request for Data Integrity Verification:**

   Cloud user sends a request with the set of encrypted HTs $\{\partial\}$ to the TPA group for verification of the data file F. In TPA group, one TPA shares and distributes this load

with other TPAs by load balancing and batch auditing methods.

2. **Challenge:**

   TPA group generates a set of random numbers $R = \{r_1, r_2.....r_i\}$ for the challenge operation. Set of these random numbers R is sent to the CSP for verification of data blocks corresponding these random numbers. The generation of these random numbers is given in the following Algorithm 4.3.

Algorithm 4.3 Challenge Operation

Input: $\{0,1\}^k$.
Output: Random numbers r_1, r_2...r_i.

1: Random numbers r_1, r_2.....r_i are generated for challenge operation.
2: $r_1 \xleftarrow{R} \{0,1\}^k$
3: $r_2 \xleftarrow{R} \{0,1\}^k$
4: $r_i \xleftarrow{R} \{0,1\}^k$

3. **Proof Generation:**

   With the help of received set R, CSP selects and retrieves the file blocks $F[r_1], F[r_2], ...F[r_i]$ for proof generation. For this purpose, CSP generates algebraic signature for these file blocks and computes their homomorphic tag. The set of these selected tags is defined as $T^{'}$. Thereafter, CSP computes encryption over these tags, which is defined as the set $\partial^{'}$. CSP sends this set $\partial^{'}$ to the TPA for verification. All these steps are described in the following algorithm 4.4.

Algorithm 4.4 Proof Generation

Input: Random file blocks $F[r_1], F[r_2], ...F[r_i]$.
Output: Set $T^{'}$, Set $\partial^{'}$.

1: **for** $int\ j = r_1;\ j \leqslant r_i;\ j++$ **do** ▷ Proof Generation
2: $T[j] = AS_g(F[j])$
3: **end for**
4: $T^{'} = \{T[r_1], T[r_2], ...T[r_i]\}$, where $T^{'}$ is the set of selected homomorphic tags.
5: **for** $int\ h = T[r_1];\ h \leqslant T[r_i];\ h++$ **do**
6: $\partial_h = E_{k_t}(T[h])$
7: **end for**
8: $\partial^{'} = \{\partial_1, \partial_2, ..., \partial_i\}$, where $\partial^{'}$ is the set of encrypted homomorphic tags.

4. **Proof Verification:** TPA group is also responsible for the proof verification. For this

purpose, the TPA group decrypts sets ∂ and $\partial^{'}$ by employing the tag decryption key k_t. Then, TPA group estimates the sets T and $T^{'}$, and then verifies whether they are equal or not. If it is equal, it indicates that the integrity of the file is preserved else it is corrupted. These steps are given in Algorithm 4.5.

Algorithm 4.5 Proof Verification

Input: Private Decryption key of TPA group k_t, Decryption function D_{k_t}, sets ∂ and $\partial^{'}$.
Output: Verification Result $T \stackrel{?}{=} T^{'}$.

1: **for** $int\ j = r_1;\ j \leqslant r_i;\ j++$ **do** ▷ Proof Verification
2: $T_j = D_{k_t}(\partial_j)$.
3: **end for**
4: $T^{'}$ = $\{T_{r_1}, T_{r_2}, ...T_{r_i}\}$
5: **for** $int\ j = 1;\ j \leqslant n;\ j++$ **do**
6: $T_j = D_{k_t}(\partial_j)$.
7: **end for**
8: T = $\{T_1, T_2, ...T_n\}$
9: Verifies $T \stackrel{?}{=} T^{'}$.

5. **Result Notification:** TPA group notifies the final result to the cloud user.

4.2.3 Dynamic Data Operations

In this part, we describe the reinforcement of dynamic data operation through our scheme. In cloud environment, the users data is stored at different CSPs. It does not store the data at users end. Therefore, it is a desperate challenge for a cloud user to implement dynamic data operations such as update, insert, append and delete at data blocks of files to change the content. Yet another challenge is to maintain the data integrity and have assurance of careful handling of data.

For any dynamic data operation, cloud user, first, needs to generate corresponding resulted file blocks. He/She also needs to change into corresponding homomorphic tags to accommodate changes in data blocks, because this ensures reflection of changes in data blocks at the CSP domain.

Batch Auditing protocol will be successfully carried out after simple changes in homomorphic tags. In other words, successful changes in homomorphic tags ensure CSP to correctly execute processing of dynamic data operation request. Otherwise, CSP would detect

server misbehaviour with high probability as described in equation 4.13.

$$P_X = 1 - \{\frac{n-r}{n}.\frac{n-1-r}{n-1}.\frac{n-2-r}{n-2}....\frac{n-c+1-r}{n-c+1}\} \tag{4.13}$$

where, P_X indicates the probability of detection of server misbehaviour that depends on total number of file blocks n, deleted blocks r, and challenged blocks c.

A simple, straight forward and trivial way to support this operation is to download all the data from CSP and recompute homomorphic tags. But, this process is highly inefficient. Here, we demonstrate our scheme for efficiently performing dynamic data operations in cloud storage. The illustrative diagram is shown in Fig. 4.2.

1. **Update Operation:** In the cloud environment, a cloud user may require to modify some data blocks stored in the CSP server, from its current value c_{ij} to a new value $c_{ij} + \Delta c_{ij}$. This operation is called as "update operation". As defined with CBC set system, file blocks are distributed among m servers. Also accessing k blocks from total n blocks requires at max t block retrieval from each server. Hence, a user can generate updated blocks by using Δc_{ij}, without involving any other unmodified blocks. Therefore, for update operation, user constructs a change matrix ΔC as follows.

$$\Delta C = \begin{pmatrix} \Delta c_{11} & \Delta c_{12} & \cdots & \Delta c_{1m} \\ \Delta c_{21} & \Delta c_{22} & \cdots & \Delta c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta c_{n1} & \Delta c_{n2} & \cdots & \Delta c_{nm} \end{pmatrix}$$

   It may be noted that unmodified blocks are represented by zero elements in ΔC and when very small change occurs within update operation, then ΔC will be a sparse matrix. To maintain the modified and unmodified data blocks before the commit (final update) operation, the CSP finds out an updated matrix ΔU by multiplying change matrix (ΔC) with Incidence matrix I of CBC dual set system. This provides dynamic data change information on the modified data blocks. The update matrix (ΔU) is an $n \times n$ matrix,where

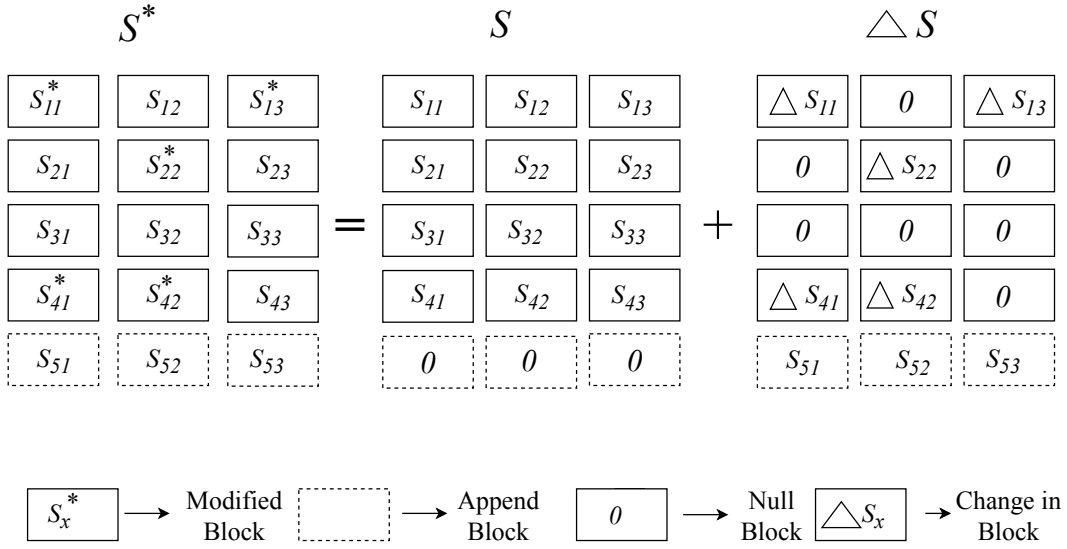


Figure 4.2: Dynamic Data Operations in DIV Method 1

$$\Delta U = \Delta C \times I$$

$$\Delta U = \begin{pmatrix} \Delta u_{11} & \Delta u_{12} & \cdots & \Delta u_{1n} \\ \Delta u_{21} & \Delta u_{22} & \cdots & \Delta u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta u_{n1} & \Delta u_{n2} & \cdots & \Delta u_{nn} \end{pmatrix}$$

Here, Δu_{ij} denotes the dynamic data change information, and (i, j) denotes the block number.

Data update operation affects some or all the modified blocks. Therefore, a change or amendment in corresponding homomorphic tag is required for each modified blocks. For every modified blocks, CSP needs to exclude occurrence of old block and replace it with new modified block. With the help of our homomorphic tag generation, this amendment has been performed efficiently only for modified blocks and not for all blocks. After the commit (final update) operation, CSP performs "final update operation" on the modified data blocks according to dynamic data change information and notifies user the completion of update operation as $c_{ij} \leftarrow c_{ij} + \Delta u_{ij}$; for all $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$.

2. **Delete Operation:**

   In some situations, after saving data on cloud, user may need to delete some blocks. We consider this "delete operation" as a special case of "update operation", in which original data blocks are replaced with the blocks that are filled with zeros or any special predetermined symbols. Thus, for "delete operation", we change the "update operation" setting Δc_{ij} in ΔC to be as $-\Delta c_{ij}$. The remaining process is same as "update operation".

3. **Append Operation:** In some conditions, user may want to add data to the stored file by adding additional data blocks at the end of the file. We refer this operation as "append operation". In cloud environment, "append operation" occurs very often. This process is bulky, because most of the users try to add large number of file blocks at any point of time.

   To support this operation, we concatenate additional blocks at the end of the corresponding file blocks in CBC matrix *C* of file F. This is equivalent to the appending blocks towards the end of data file.

   If we assume that initially n number of rows in CBC matrix *C* and user wants to append j number of blocks at the end of file F, the file blocks are represented as $(F[n+1], F[n+2], ..., F[n+j])$.

   To start the append operation, CSP performs zero padding for each server m and creates a new CBC matrix C_{new} from *C* with j file blocks. Then, CSP distributes these file blocks among m servers, which randomly stores these blocks by load balancing. After this, CSP needs only to perform "update operation" for this additional $(F[n+1], F[n+2], ..., F[n+j])$ file blocks in C_{new}.

   These additional file blocks $(F[n+1], F[n+2], ..., F[n+j])$ are added into the file. Therefore, CSP needs to regenerate homomorphic tag. For this purpose, CSP performs operation the operation of homomorphic tag generation for these additional file blocks along with the previous $F[n]$ blocks.

   Finally, CSP notifies the user at the completion of "append operation" as $C_{new} \leftarrow C_{new} + \Delta u_{new}$; for all file blocks $n \in \{1, ..., j\}$ with new homomorphic tag T_{new}.

   Figure 4.2 demonstrates the support of dynamic data operations (Update, Delete and Append operation) for our scheme. In this figure, Δc_{11}, Δc_{22}, Δc_{41}, and Δc_{42} are the

update information blocks of c_{11}, c_{22}, c_{41}, and c_{42} file blocks, respectively. $-\Delta c_{11}$ is the delete information of file block c_{11} and c_{51}, c_{52}, and c_{53} are the appended file blocks.

4. **Insert Operation:**

   An insert operation in a data file can be considered as a special case of append operation at a desired index position for which, CSP maintains same CBC data structure for the entire file. Thus, at index i, inserting a file block $F[i]$ requires shifting of all subsequent blocks starting from the n index position by one index slot. It affects many rows in CBC matrix *C* and requires a number of substitution computations. This also needs re-indexing of all subsequent blocks and regeneration of homomorphic tags.

   To support insert operation, existing approaches [19, 48] requires additional data structures like Merkle Hash Tree [93] and RSA Tree [48] to maintain data index information. This additional information is stored and maintained at the user end, which is an overhead for end user.

   In our approach, insert operation at a specific index is easy and efficient because CSP does not require any additional data structure to maintain data index information and causing no extra overhead at user end. The same CBC structure is utilized for this purpose.

4.2.4 Proof of Batch Auditing

In this subsection, we provide the proof of batch audit session through valid evidence. With batch auditing, TPA group is able to take various audit tasks on distinct data files from multiple users which is beneficial for the end user as TPA group performs multiple tasks together at an instance of time. Figure 4.3 shows our batch auditing technique. For better understanding of batch auditing, we take a simple example of C cloud users, which have j multiple audit tasks on k number of files.

1. **Set Up Phase:**

   From C users, each user c has a data file F_c = $\{F_1^c, F_2^c, ...F_n^c\}$ to outsource on the cloud server, where $c \in \{1, ..., C\}$. To simplify, we assume that each file has n numbers of blocks and each user c has chosen a random number to generate

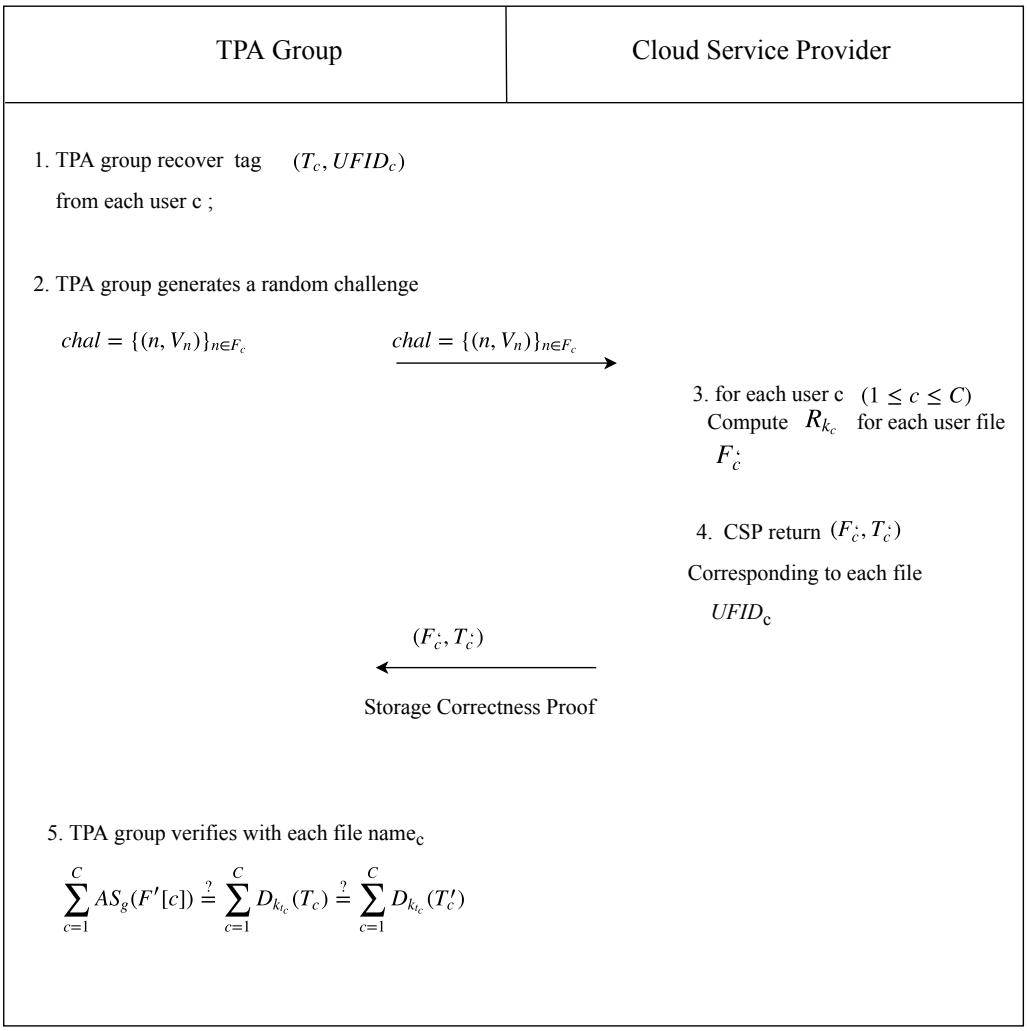


Figure 4.3: Batch Auditing Process in DIV Method 1

encryption key for TPA group. We also assume that each user file F_c randomly selects a Unique File Identifier $UFID_c \in \mathbb{Z}_{\mathbb{P}}$.

Thus, for each user c, CSP denotes his / her TPA group key as (k_{tc}) and corresponding file tag as ($UFID_c$, k_{t_c}). Now, for each user c, CSP runs homomorphic tag generation algorithm, computes $AS_g(F_n^c)$, and sends (T_c, $UFID_c$) to TPA group.

2. **Audit Phase:** TPA group recovers tag (T_c, $UFID_c$) from each user c and sends audit challenge *chal* = $\{(n, V_n)\}_{n \in F_c}$ to the CSP server for auditing data files of all C users. After successful receiving of *chal* from each user $c \in \{1, ..., C\}$, CSP randomly chooses $r_k \in \mathbb{Z}_{\mathbb{P}}$ and computes R_{k_c} for each user file F_c' using the following procedure.

For $\{0 < j \leq R_{k_c}\}$

$l_j = \sigma_{k_{t_c}}(r_k + j)$

$F'[c] = F'[c] + F[l_j]$

CSP returns $(F'[c], T'_c)$ corresponding to each file $UFID_c$. Now, TPA group verifies each file $UFID_c$, whether equation 4.14 holds or not.

$$\sum_{c=1}^{C} AS_g(F'[c]) \stackrel{?}{=} \sum_{c=1}^{C} D_{k_{t_c}}(T_c) \stackrel{?}{=} \sum_{c=1}^{C} D_{k_{t_c}}(T'_c) \tag{4.14}$$

We can expend the Left-Hand Side (LHS) of the equation 4.14 as :

$$\begin{aligned}
\Rightarrow \quad & \sum_{c=1}^{C} AS_g(F'[c]). \\
\Rightarrow \quad & AS_g(k_{t_1}, UFID_1) + \sum_{c=1}^{C-1} AS_g(F'[c]). \\
\Rightarrow \quad & D_{k_{t_1}}(T_1) + \sum_{c=1}^{C-1} AS_g(F'[c]). \\
\Rightarrow \quad & D_{k_{t_1}}(T_1) + D_{k_{t_2}}(T_2) + \sum_{c=1}^{C-2} AS_g(F'[c]). \\
\Rightarrow \quad & \sum_{c=1}^{C} D_{k_{t_c}}(T_c) .
\end{aligned}$$

We can also easily find out the third equality of the equation (4.14) with properties of algebraic signature and CBC.

$$\begin{aligned}
\Rightarrow \quad & \sum_{c=1}^{C} AS_g(F'[c]). \\
\Rightarrow \quad & AS_g(k_{t'_1}, UFID_1) + \sum_{c=1}^{C-1} AS_g(F'[c]). \\
\Rightarrow \quad & D_{k_{t_1}}(T'_1) + \sum_{c=1}^{C-1} AS_g(F'[c]). \\
\Rightarrow \quad & D_{k_{t_1}}(T'_1) + D_{k_{t_2}}(T'_2) + \sum_{c=1}^{C-2} AS_g(F'[c]). \\
\Rightarrow \quad & \sum_{c=1}^{C} D_{k_{t_c}}(T'_c) .
\end{aligned}$$

Hence, it is proved that our scheme supports batch auditing with multiple TPA in cloud environment.

4.2.5 Efficiency Improvement

As we have shown in equation 4.14, TPA group can able to perform auditing tasks simultaneously with batch auditing. We also observe that it reduces computation cost on TPA group end, because aggregating of C verification equations into one equation helps to reduce the number of expensive operations into $C+1$ from $2C$, as required in single TPA auditing. This saves a precious amount of auditing time. Hence, it improves the auditing efficiency.

4.3 DIV Method 2: Using PHC System and CBC

In this section, first, we describe the system model to apply for DIV method 2. Then, we explain the DIV technique with Paillier Homomorphic Cryptography system (PHC) [90]. Thereafter, dynamic data operations and proof of batch auditing are given for the utility description of this method.

4.3.1 System Model

In this work, we propose a three-tier architecture for privacy preserve auditing. The first tier is cloud user, the second tier is TPA, and the third tier is cloud service provider. The details of each tier is discussed under the following heads.

1. **Cloud Users**

   Any cloud service customer can be treated as a cloud user who has limited resources to perform the DIV process. User requires additional resources to do batch auditing. Thus, user delegates batch auditing to the TPAs due to limited resources .

2. **Third Party Auditor (TPA)**

   TPA is the trusted entity. The cloud users assign the DIV task to the trusted TPAs. TPA performs batch audit task assigned from the users.

3. **Cloud Service Provider(CSP)**

   CSPs are the firms which have enough infrastructure and resources to provide the services to the cloud users. CSP has two components: (1) Processing server and (2) Data storage. Processing server is used to process the user's request. Data storage is used to store the encrypted data and assessment keys for the convenience of cloud

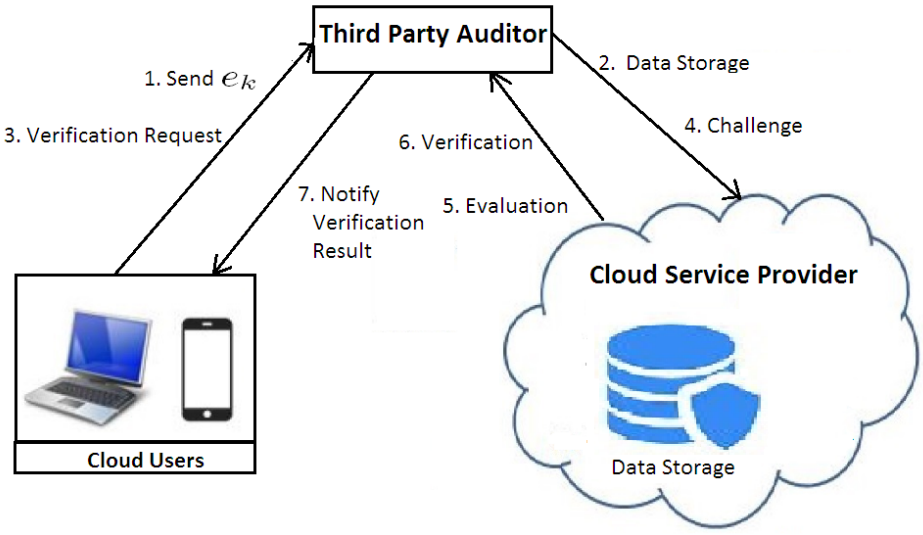


Figure 4.4: Workflow of the Proposed Privacy Auditing DIV Method 2

users.

4.3.2 Workflow of the Proposed Method

In this subsection, we present the process flow of our proposed audit scheme. Let us assume that a cloud user has a file F which consists of n blocks. These n file blocks are represented as $F[1], F[2], ...F[n]$. The cloud user wants to store this file in CSP.

In the first step, a cloud user generates encryption and decryption keys for the file in key generation process. Next, the user encrypts the file with multi-power RSA algorithm [90,94] using encryption process and sends it to CSP for storing the encrypted file. CSP stores the encrypted file in its data storage. When a user requests processing server to assess the file, the CSP performs the assessment process and returns the encrypted response to the TPA. TPA decrypts the encrypted response. Finally, TPA forwards the verification results to the cloud user. The workflow of the three-tier architecture is shown in Figure 4.4. The detail descriptions of these tasks are given below.

1. **Key Generation**

   In this step, the cloud user induces two keys, the assessment key (A_k) and the secret key (ϕ_k) at the client side. These keys are generated using a variant of Paillier Cryptography System [90, 94]. The key generation algorithm takes two inputs n and

r as parameters and generates two distinct primes *x* and *y*. Both numbers (*x* and *y*) are $[\frac{n}{r+1}]$ bits long and are used to generate the keys. We have selected an integer *e* which is relatively prime to *T* and *x*. This integer is used to generate secret key. In Algorithm 4.6, the generation process of key pairs is demonstrated.

Algorithm 4.6 Key Generation

Input: n, r .
Output: Assessment key (A_k), The secret key is ϕ_k.

1: Compute $\lambda = x^r y$ and *T* =(x - 1)(y- 1).
2: Compute $d \equiv (e^{-1}) mod\ T$.
3: Compute $d_x \equiv (d) mod\ x$ and $d_y \equiv (d) mod\ y$.
4: Compute $k_x = \frac{\lambda}{x^r}(\frac{\lambda}{x^r})^{-1}$; $k_y = \frac{\lambda}{y}(\frac{\lambda}{y})^{-1}$; where $k_x \equiv (1) mod\ x^r$ and $k_y \equiv (1) mod\ y$.
5: Compute the assessment key is A_k= (λ); the secret key is ϕ_k =(λ; x; y; r; e; d_x; d_y; k_x; k_y).

2. **Storage**

   The cloud user breaks the file *F* into a number of file blocks $F[1], F[2], ...F[n]$. The size of each file block is $512MB$. The cloud user encrypts these blocks with the secret key ϕ_k using RSA technique [91] and Euler's totient function [94]. After that user sends the encrypted data and the assessment key A_k to the CSP for storage. If we denote $F[i] \in \mathbb{Z}_\lambda$ be a plaintext for the i^{th} block and the secret key ϕ_k, then ciphertext $S[i]$ can be calculated using equation 4.15

$$S[i] \equiv (F[i]^e) mod\ \lambda \tag{4.15}$$

   At the CSP server, file blocks are presented in encrypted form as assessment key (A_k) individual ciphertexts $(S[1]; S[2]; S[3]; ::: S[n])$. These encrypted blocks and the assessment key (A_k) are stored on the cloud data storage server for verification purpose.

3. **Verification Request**

   User sends a verification request to the TPA when he wants to verify the stored data. Now, TPA does the further processing.

4. **Challenge**

   The TPA searches and recomputes the individual block information from the cipher-

texts of each block using the assessment process, which is described in the next step.

5. **Assessment**

   In this process, the CSP server recomputes overall ciphertext from the ciphertext of each block. To do this, it performs cross product of the ciphertext for each block. We denote overall ciphertext as $\mathbb{S}$ and the ciphertext of each block as $((S[1]; S[2]; S[3]; ::: S[n]))$.

   For this, the processing server needs to retrieve and verify the encrypted data $(S[1]; S[2]; S[3]; ::: S[n])$. To do this, CSP provides $\mathbb{S}$ information. Now, ciphertext of each block is computed using F_i^e. It recomputes the cross product of individual block information $(F[1] \times F[2] \times F[3] \times ::: \times F[n])^e \; mod \; N$ by using the assessment key A_k. Now, CSP will be able to assess the overall cipertext $\mathbb{S}$. This procedure is mathematically represented in equation 4.16

$$\begin{aligned} &(F[1]^e \times F[2]^e \times F[3]^e \times ..... \times F[n]^e) mod \; \lambda \\ &\equiv (F[1] \times F[2] \times F[3] \times ::: \times F[n])^e mod \; \lambda \\ &\equiv E(\phi_k, F[1] \times F[2] \times F[3] \times ::: \times F[n]) \\ &\equiv \mathbb{S}. \end{aligned} \tag{4.16}$$

   Thus, $\mathbb{S}$ is generated by using equation (4.17).

$$\mathbb{S} = Assessment(A_k; S[1]; S[2]; S[3]; .....S[n]) \tag{4.17}$$

6. **Verification**

   The cloud service provider returns the processed result in encrypted form to TPA. Now, the user can evaluate and verify the plaintext $\mathbb{F}$ using decryption method.

   For the given secret key ϕ_k, TPA utilizes the Hensel lifting theorem [91] to decrypt the ciphertext $\mathbb{S}$ and constructs the plaintext. For this operation, modulus of F_x^e is performed with respect to x^r. This is done for all values from 1 to $r-1$, where r is a random number which is defined at the time of key generation. After this, a user utilizes the Chinese remaindering theorem to recover the plaintext. To do this, TPA calculates $(F_x) mod \; x^r$ and $(F_y) mod \; y$ that alternatively generates $\mathbb{F}$. The details of Hensel Lifting theorem (Theorem 4.1.2) and Chinese Remaindering theorem (Theo-

rem 4.1.4) is described in subsection 4.1.6. The steps to decryption of cipher-texts are summarized in Algorithm 4.7.

Algorithm 4.7 Decryption

Input: a private key $P_k = (\lambda; x; y; r; e; d_x; d_y; k_x; k_y)$ and a ciphertext $\mathbb{S} \in \mathbb{Z}_\lambda$.
Output: $\mathbb{F} \in \mathbb{Z}_\lambda$.

1: Compute $S[1] \equiv (S) mod\ x$; $S[x] \equiv (S) mod\ x^r$ and $S[y] \equiv (S) mod\ y$.
2: Compute $F[1] \equiv ((S[1])^{d_x})\ mod\ x$ and $F[y] \equiv ((S[y])^{d_y})\ mod\ y$.
3: Using the Hensel lifting, we construct a plaintext $F[x]\ modulo\ x^r$ such that $S[x] \equiv ((F[x])^e) mod\ x^r$. To do this, first we set $K[1] \leftarrow F[1]$.
4: int i=2;
5: **while** $i \leq r$ **do**
6: $E[i] \leftarrow ((K[i])^e) mod\ x^{i+1}$.
7: $N[i] \leftarrow (S[x] - E[i]) mod\ x^{i+1}$.
8: $G[i] \leftarrow N[i]/x^i$.
9: $F[i] \leftarrow G[i] \times ((eM_0^{e-1})^{-1}) mod\ x$.
10: $K[i] \leftarrow K[i-1] + x^i F[i]$.
11: **end while**
12: Using the Chinese remaindering, recover the plaintext $\mathbb{F} = F[1] \times F[2] \times F[3] \times .... \times F[n]$ such that $F \equiv (F[x]) mod\ x^r$ and $F \equiv (F[y]) mod\ y$. To do this, first set $F[x] \leftarrow K[r]$.
13: Compute $\mathbb{F} \leftarrow (F[x] \times k[x]) + (F[y] \times k[y]) mod\ \lambda$.
14: Output: $\mathbb{F} \in \mathbb{Z}_\lambda$.

7. **Notify the Verification Result**

   Finally, TPA notifies the verification result to the cloud user, whether the integrity of data is maintained or not.

4.3.3 Correctness Proof of Privacy Preservation

In this subsection, we prove the correctness of our approach for the effective privacy preservation. Let, $\lambda = x^r y$, where x, y be two distinct primes and $r \geq 2$. We consider two integers e and d such that $e.d \equiv (1)\ mod (x-1)(y-1)$ and $gcd(e, x) = 1$.

Next, we assume the threat vector r. Here, we select $r \geq 2$ for making the factorization difficult. For this value of r, any crypto-logical system fails to decrypt the information from the stored data.

In our approach, there are three way to reduce the cipher-text. First, $S[y]$ is reduced with modulo y. In other way, $S[1]$ can be reduced with modulo x. Final way to reduce $S[x]$ is with modulo x^r. For further reduction of d_y and d_x, we have relations $d_y \equiv (d)\ mod\ (y-1)$

and $d_x \equiv (d)\ mod\ (x-1)$. If there exists two integers $n_1, n_2 \in \mathbb{Z}$, we can use both the integers to reduce the relationship using equations 4.18 and 4.19.

$$d = d_y + n_2\ (y-1) \tag{4.18}$$

$$d = d_x + n_1\ (x-1) \tag{4.19}$$

From equation (4.15), we get the ciphertext $\mathbb{S} \equiv (F^e) mod\ \lambda$ where $F \in \mathbb{Z}_\lambda$. Now, we decrypt $\mathbb{S}$ modulo x as

$$\begin{aligned} F[1] &\equiv (S^d) mod\ x \\ &\equiv (S[1] + bx^d) mod\ x; \quad where\ b \in \mathbb{Z} \\ &\equiv ((S[1])^{d_x + a_1(x-1)})\ mod\ x \\ &\equiv ((S[1])^{d_x})\ mod\ x \quad [Thanks\ to\ Theorem\ 4.1.6] \end{aligned} \tag{4.20}$$

Similarly, we can have, $F[y] \equiv ((S[y])^{dy}) mod\ y$ for the second method.

If we choose the plaintext as $F[x]$ and the ciphertext as $S[x]$, relation between them can be represented as follows: $S[x] \equiv (F[x])^e\ mod\ (x^r)$. The x-adic expansion of $F[x]$ is given as:

$F[x] \equiv F[1] + x.F[2] + x^2.F[3] + ... + x^{r-1}.F[r] mod\ x^r$, where $F[1]; ....; F[r] \in \mathbb{Z}_x$.

If we select a function $G[i](F[1]; F[2]; ....; F[i+1])$, then for $0 \leq i \leq r$

$S[x] \equiv G[i](F[1]; F[2]; ....; F[i+1]) \equiv (F[1] + x.F[2] + x^2.F[3] + ... + x^i.F[i+1])^e$;

With the help of binomial Theorem 4.1.3, we can reduce $S[x]\ modulo\ x^{i+1}$

$G[i](F[1]; F[2]; ....; F[i+1]) \equiv (F[1] + x.F[2] + x^2.F[3] + ... + x^i.F[i+1])^e mod\ x^{i+1}$

$\equiv \sum_{k=1}^{e} \binom{e}{k} (F[1] + x.F[2] + x^2.F[3] + ... + x^{i-1}.F[i])^(e-k)(x^i F[i+1])^k mod\ x^{i+1}$

Now, we can explore this equation with the following expansion:

$\equiv (F[1] + x.F[2] + x^2.F[3] + ... + x^{i-1}.F[i])^e + (F[1] + x.F[2] + x^2.F[3] + ... + x^{i-1}.F[i])^{(e-1)} e x^i F[i+1] + \sum_{k=2}^{e} \binom{e}{k} (F[1] + x.F[2] + x^2.F[3] + ... + x^{i-1}.F[i])^{(e-k)} (x^i F[i+1]^k mod\ x^{i+1}$

The above expansion can be divided into two sub-parts expressed as follows:

$\equiv (F[1]+x.F[2]+x^2.F[3]+....+x^{i-1}.F[i])^e+ex^iF[i+1]\times\sum_{j=1}^{e}\binom{e}{j}F[1]^{e-j}\times(x.F[2]+x^2.F[3]+....+x^{i-1}.F[i])^j mod\ x^{i+1}$

$\equiv (F[1]+x.F[2]+x^2.F[3]+....+x^{i-1}.F[i])^e+x^ieF[1]^eF[i+1] mod\ x^{i+1}$

It is observe that for $1 \leq i \leq r$; $G[i-1] \equiv (F[1]+x.F[2]+x^2.F[3]+....+x^{i-1}.F[i])^e$

Thus, $S[x] = G[i-1]+x^ieF[1]^eF[i+1] mod\ x^{i+1}$

We have found the recursive relation for the values $F[1]; F[2]; ....; F[r]$ in Eq 4.21.

$$F[i] \equiv (\frac{S[x]-G[i-1](mod\ x^{i+1})}{x[i]})\times e^{-1}F[i]^{1-e} mod\ x \tag{4.21}$$

In this equation 4.21 , x and $F[1] \in \mathbb{Z}_x$ are relative prime with e. Thus, we can see that $F[1]^{e-1}$ and e are the reverse modulo of x. From this proof, it is evident that our approach correctly supports the privacy preservation.

4.3.4 Support for Dynamic Data Operations

Our approach supports three dynamic data operations, which are described as follows:

4.3.4.1 Update Operation

In update operation, the existing file blocks are modified to restructure the entire file. The first homomorphic property of a PHC is used to do this update operation. According to this property, if we take product of two cipher-texts and subsequently decrypt this product, then this is equivalent to the sum modulo of corresponding plain-texts.

In this work, two plain-text file blocks are represented as $F[x]$, $F[y]$ and two random numbers are denoted by n_1, n_2 such that n_1, $n_2 \in \mathbb{Z}_\lambda^*$. The following equation describes the first property of PHC.

$$D[E(F[x],n_1).E(F[y],\ n_2)] = D[E(F[x]+F[y],\ n_1.n_2) \mod \lambda^2] = F[x]+F[y] \mod \lambda \tag{4.22}$$

With this property, CSP can perform the dynamic update operation. We do not need to retrieve the original plain-texts. We only require to perform the product of two cipher-texts

and subsequently decrypt this product. As a result, we get the sum modulo of two plain-texts.

To update a file block $F[i]$, a user modifies this block with small change $\pm\Delta F[i]$. It creates a new block $F[i] \pm \Delta F[i]$. Thus, change in ciphertext will be $(F[i] \pm \Delta F[i])^e$. It results the final ciphertext as $S[i] \pm \Delta S[i]$, after the commit operation.

4.3.4.2 Append Operation

In an append operation, new data blocks are added at the end block of the file. This operation increases the size of the stored data. We use the second homomorphic property of PHC to perform this operation. According to this property, if we take the raised power of one ciphertext with another cipher-text, and subsequently decrypt this result, then the result is equivalent to the product modulo of two plain-texts.

In our notations, two file blocks are represented as $Fx, F[y] \in \mathbb{Z}_\lambda$ and two random numbers are denoted by n_1, n_2 such that n_1, $n_2 \in \mathbb{Z}_\lambda^*$.

The equations 4.23 and 4.24 describe the second property of PHC.

$$D[E^{F[y]}(F[x], n_1)] = D[E(F[x].F[y],\ n_1^{F[y]}) \mod \lambda^2] = F[x].F[y] \mod \lambda \quad (4.23)$$

$$D[E^{F[x]}(F[y], n_2)] = D[E(F[x].F[y],\ n_2^{F[x]}) \mod \lambda^2] = F[x].F[y] \mod \lambda \quad (4.24)$$

With this property, we can perform the dynamic append operation and we do not need original plain-texts. We only need to do the raised power of one cipher-text with another cipher-text and subsequently decryption of this result. In output, we get the product modulo of two plain-texts.

4.3.4.3 Delete Operation

In delete operation, some data blocks are deleted. It is a special case of update operation, in which some data blocks are replaced with null blocks.

There are two types of delete operation: (1) Partial block delete operation, and (2) Full block delete operation. In partial block delete operation, CSP retrieves the file from the

storage server in encrypted form and some part of blocks are deleted, and modified blocks are stored on the server. In full block delete operation, CSP retrieves the file from the storage server in encrypted form, and the full blocks of the file are replaced with null blocks. These null blocks are assigned to the scheduler for the fresh allocation.

4.3.5 Use Case

We have considered a use case to show the different dynamic operations performed by a user in the data file.

1. **Update Operation:**

   Cloud user requests to CSP for the update operation in a file. CSP retrieves this file from the storage server in encrypted form and uses the first homomorphic property of a Paillier Cryptography System as outlined in equation 4.22. This property enables a server to update the file. Pictorial representation of update operation is given in Figure 4.5. Here, we represent data blocks of a file as the *S* matrix and the actual file block as $S_{11}, S_{13}, S_{22}, S_{41}, \ and \ S_{42}$. If the modification or updation in file block are described as $\triangle S_{11}, \triangle S_{13}, \triangle S_{22}, \triangle S_{41}, and \ \triangle S_{42}$, then $S^*_{11}, S^*_{13}, S^*_{22}, S^*_{41}, \ and \ S^*_{42}$ are depicted as updated output blocks.

2. **Append Operation:**

   Cloud user requests to CSP for the append operation in the file. CSP retrieves this file from the storage server in encrypted form and uses the second homomorphic property of a Paillier cryptography system as outlined in equations 4.23 and 4.24. This property enables the server to append the data to the file. Pictorial representation of append operation is given in Fig. 4.5. In this figure, we represent data blocks of a file as the *S* matrix and the appended file block as $S_{51}, S_{52}, \ and \ S_{53}$. After the commit operation, these blocks are added in file matrix S^*.

3. **Delete Operation:**

   Cloud user requests to CSP for the delete operation in a file. In partial delete operation, only some partial block information are to be deleted, and blocks are to be converted into the modified blocks. The modified blocks co-exist after the commit operation on blocks. We represent data blocks of a file as the *C* matrix. The explana-

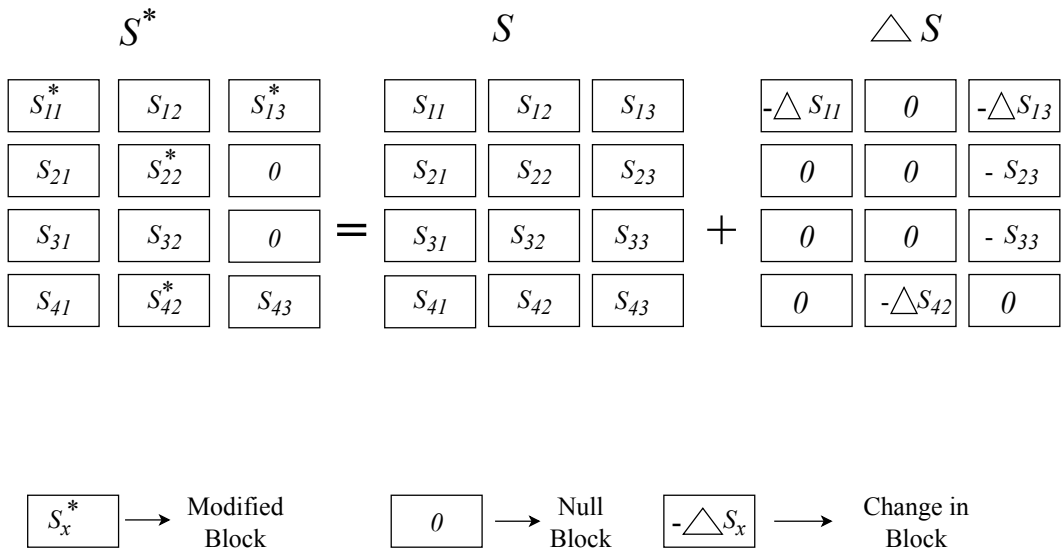


Figure 4.5: Dynamic Update and Append Operation in DIV Method 2

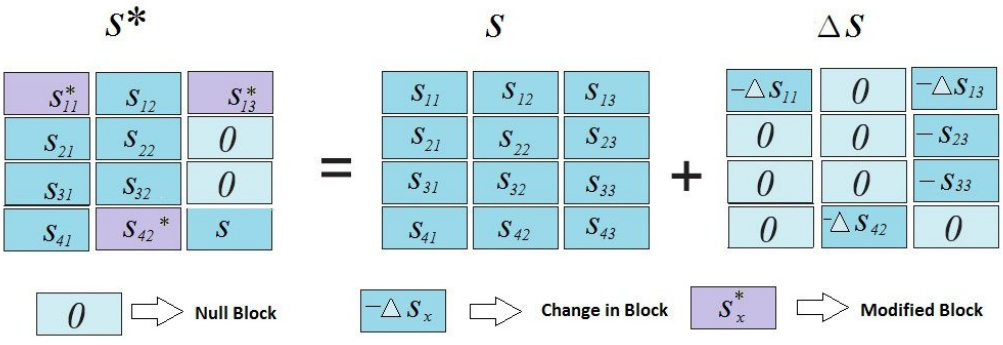


Figure 4.6: Dynamic Delete Operation in DIV Method 2

tion of delete operation is shown in Figure 4.6.

In Figure 4.6, if three blocks such as $S_{11}, S_{13}, \ and \ S_{42}$ are selected for the partial delete operation, and $-\triangle S_{11}, -\triangle S_{13}, \ and \ -\triangle S_{42}$ are information to be deleted from the respective blocks. Then, the modified blocks are depicted as the block $S^*_{11}, S^*_{13}, \ and \ S^*_{42}$.

In full block delete operation, entire blocks are to be deleted. After deletion of full blocks, blocks are replaced with the null blocks. Finally, at the time of commit operation, these null blocks are allocated as a fresh data blocks to the scheduler, which can be assigned for storing the information.

In Figure 4.6, if two blocks such as S_{23} and S_{33} have selected for the full delete operation, and $-S_{23} \ and \ -S_{33}$ are the block change, then the modified blocks are represented by null blocks as shown in Figure 4.6 which can be assigned to the scheduler as a fresh data blocks.

4.4 Experiments

To perform the experiments related to DIV method 1 and DIV method 2, we have used two Personal Computers(PC's) configured with Intel Core i7-2600S 2.80 GHz and 16 GB RAM. We have used one PC as a TPA. For this purpose, we configured Cloudera CDH 5.3.0-0 [95] on this PC. For file storage, we have used another PC, which is configured with Citrix Xen Server 6.2.0 [96]. We have tested the DIV method 1 and 2 with different performance parameters. The detailed discussion of experimental result is given in chapter 6.

4.5 Summary

In this chapter, we proposed two collaborative techniques for data integrity verification with multiple third-party auditors. The first technique utilises the algebraic signature, homomorphic verification tag, and combinatorial batch codes while the second technique utilises the Paillier homomorphic cryptography system for data integrity verification. Properties of algebraic signature and CBC demonstrate the suitability for data integrity verification in cloud computing. On the other hand, the properties of the Paillier homomorphic cryptography

system make the second approach very helpful in performing data integrity verification in a secure manner in cloud environments. We achieve an average accuracy of 95% and 92.76% for CBC based and Paillier homomorphic cryptography system based data integrity verification methods, respectively. Further, both the approaches support dynamic data operations with adaptable and useful batch auditing through which various audit sessions for multiple users can be handled simultaneously.

Chapter 5

Distributed Denial of Service (DDoS) Attack Prevention

Distributed Denial of Service (DDoS) attack [77] is a customize version of DoS attack which works in collaborative manner on the availability and functionality of the targeted cloud. Multiple corrupted systems are used for targeting and damaging a victim cloud to build a DDoS attack. After stealing the control, an associate intruder imposes these compromised systems on sending a large number of malicious packets on the targeted cloud. In this distributed approach, multiple systems are used by the attacker to launch DoS attack. In DDoS attack, multiple compromised systems of cloud act as victim systems. These victim systems are caused to damage other systems and increase the potential of attack on the targeted system. In this chapter, we present a third party based technique to prevent DDoS attack in cloud environment.

In section 5.1, we discuss the preliminaries of Dempster Shafer Theory and Weibull Probability Distribution which are used in our proposed work. Section 5.2 provides the proposed system model for the DDoS attack prevention in cloud environment. Description of the proposed technique is given in section 5.3. Finally, summary of this work is presented in section 5.4.

5.1 Preliminaries

In this section, first we introduce the Dempster Shafer Theory (DST) and Weibull Probability Distribution, which are used in the proposed DDoS prevention method.

5.1.1 Dempster Shafer Theory (DST)

DST [97], [98] is a powerful method for mathematical diagnostics, statistical inference, decision analysis, and risk analysis. In DST, probabilities are assigned on mutually exclusive elements of the power set of state space (Ω) (all possible states). This state space (Ω) is also called as frame of discernment, and the assignment procedure of probabilities is called basic probability assignment *(bpa)*.

According to DST method [99] for a given state space (Ω), the probability (called mass) is assigned for t sets of all 2^{Ω} elements, which are all possible subsets of (Ω). DST works with 3-valued logic, which provides Fault Tree Analysis (FTA) [98]. For example, if a standard state space (Ω) is (True, False), then 2^{Ω} should have 4 elements: {Φ, True, False, (True, False)}. The (True, False) element describes the imprecision component, which is introduced by DST. These elements refer to the value either true or false, but not both. The sum of all probabilities for DST is equal to 1 which is given in equation 5.1, and P(Φ) is 0.

$$P(True) + P(False) + P(True, False) = 1 \tag{5.1}$$

In this work, we use FTA to analyze each VM corresponding to the victim, which is perceived by a boolean OR gate. If we choose set $A = \{a_1, a_2, a_3\}$ and $B = \{b_1, b_2, b_3\}$ as an input output set, respectively. Then, Table 5.1 describes the Boolean truth table for the OR gate. From Table 5.1 we get:

$$P(A) = (a_1, a_2, a_3) = \{P(True), P(False), P(True, False)\} \tag{5.2}$$

$$P(B) = (b_1, b_2, b_3) = \{P(True), P(False), P(True, False)\} \tag{5.3}$$

Table 5.1: Boolean Truth Table for the OR Gate

		b_1	b_2	b_3
	$\vee$	T	F	(T, F)
a_1	T	T	T	T
a_2	F	T	F	(T, F)
a_3	(T, F)	(T, F)	T	(T, F)

$$P(A \vee B) = \{a_1b_1 + a_1b_2 + a_1b_3 + a_2b_1 + a_3b_2; a_2b_2; a_2b_3 + a_3b_1 + a_3b_3\} \tag{5.4}$$

Equation (5.4) can be rewritten as equation (5.5) by putting the value from equation (5.1).

$$P(A \vee B) = \{a_1 + a_2b_1 + a_3b_2; a_2b_2; a_2b_3 + a_3b_1 + a_3b_3\} \tag{5.5}$$

Finally, our solution uses Dempster's combination rule, which fuses evidences from multiple independent sources using a conjunctive operation (AND) between two *bpa's* P_1 and P_2, called the joint P_{12}, which is given in equation (5.6).

$$P_{12}(A) = \frac{\sum_{B \cap C = A} P(B)P(C)}{1 - K} \tag{5.6}$$

In equation (5.6), the factor $1 - K$ is called normalization factor, and it is used to avoid the conflict evidence entirely, when $A \neq \Phi$; $P_{12}(\Phi)$ = 0 and $K = \sum_{B \cap C = \Phi} P(B)P(C)$.

5.1.2 Weibull Probability Distribution

The Weibull probability distribution is the essential key element for operations of TPA. We used this distribution because it provides a variable reliability function. With this function, we can easily observe and detect DDoS attack in cloud environment.

The Weibull distribution [100] is a continuous probability distribution in probability theory and statistics. Mathematically, this distribution is defined by the Probability Distribution Function (PDF). The mathematical expression of the Weibull PDF with three parameters is

given in equation (5.7).

$$F(T) = \frac{\beta}{\eta}.(\frac{T-\gamma}{\eta})^{\beta-1}.e^{-(\frac{T-\gamma}{\eta})^{\beta}} \tag{5.7}$$

where F(T) ≥ 0 , T ≥ 0 or γ, $\beta > 0$, $\eta > 0$, $-\infty < \eta < \infty$, and

- β is the shape parameter, also known as the Weibull slope.
- η is the scale parameter.
- γ is the location parameter.

The Weibull Reliability function is given by:

$$R(T) = e^{-(\frac{T-\gamma}{\eta})^{\beta}} \tag{5.8}$$

The Weibull Failure Rate function is given by:

$$\lambda(T) = \frac{F(T)}{R(T)} = \frac{\beta}{\eta}.(\frac{T-\gamma}{\eta})^{\beta-1} \tag{5.9}$$

The Weibull Mean Life or Mean Time to Failure (MTTF) is given by:

$$\overline{T} = \gamma + \eta.\tau(\frac{1}{\beta} + 1) \tag{5.10}$$

Where, τ (*) is the Gamma function. The Gamma function is defined as:

$$\tau(n) = \int_0^{\infty} e^{-x}.x^{n-1}dx \tag{5.11}$$

The equation for the median life, or β_{50} life, for the Weibull distribution is given by:

$$\widetilde{T} = \gamma + \eta(ln2)^{\frac{1}{\beta}} \tag{5.12}$$

We design the implementation architecture for the security and reliability with the help of "Bathtub Curve". We observe that Weibull distribution with $\beta \prec 1$, failure rate decreases with time. It is called as early life failures with respect to time. When β close to 1 or equal to 1, Weibull distribution has a relatively constant failure time. It indicates useful life with

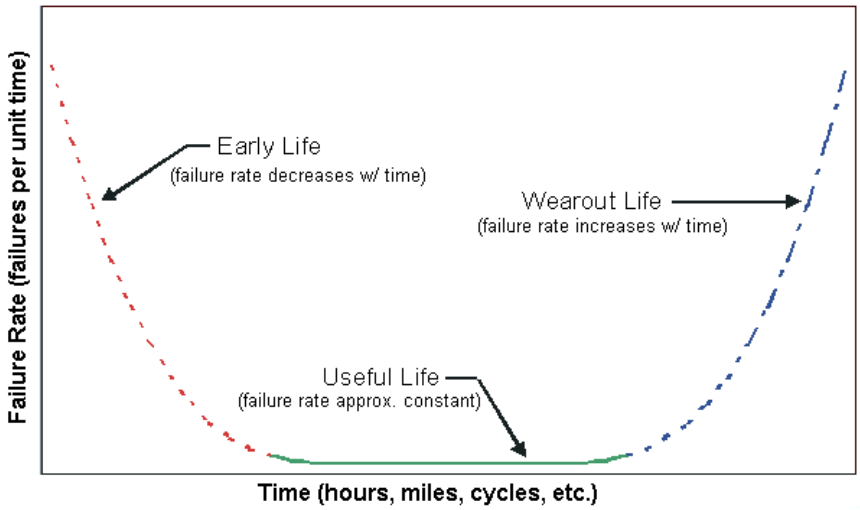


Figure 5.1: Bathtub Curve of Failure Rate

random failure. When Weibull distribution is $\beta \succ 1$, failure rate increases with time. It is also called as “Wear Out Failure". If we comprise all three condition, then it forms “Bathtub Curve” [100], which is shown in Fig. 5.1.

5.2 Proposed System Model

We propose a system model to detect and prevent the victim cloud servers from flooding attacks. In our modal, we consider a workstation as a TPA for observation of all packets reached to cloud servers. It is an independent and trustworthy entity which logs all legitimate as well as malicious packets on behalf of all cloud servers. We call this TPA as “Cloud Warrior".

In DDoS attack, there are many malicious work stations use to attack on targeted cloud storage servers. They send bulk of malicious packets on targeted machine for attack. On the other hand, some normal work stations also send legitimate packets to the targeted machine. Due to huge number of service packets, even the legitimate users are not able to obtain the services. In this condition, victim system send log details to the CW for identification of the origin source that caused the DDoS attack. Figure 5.2 shows the model representation of DDoS attack scenario with presence of Cloud Warrior (CW). In our system, when the 50% of the resources are overwhelmed (down) with DDoS attack, it sends log information to the CW. In the proposed system, the size of log information is very low (for 1 TB, it is 256kb)

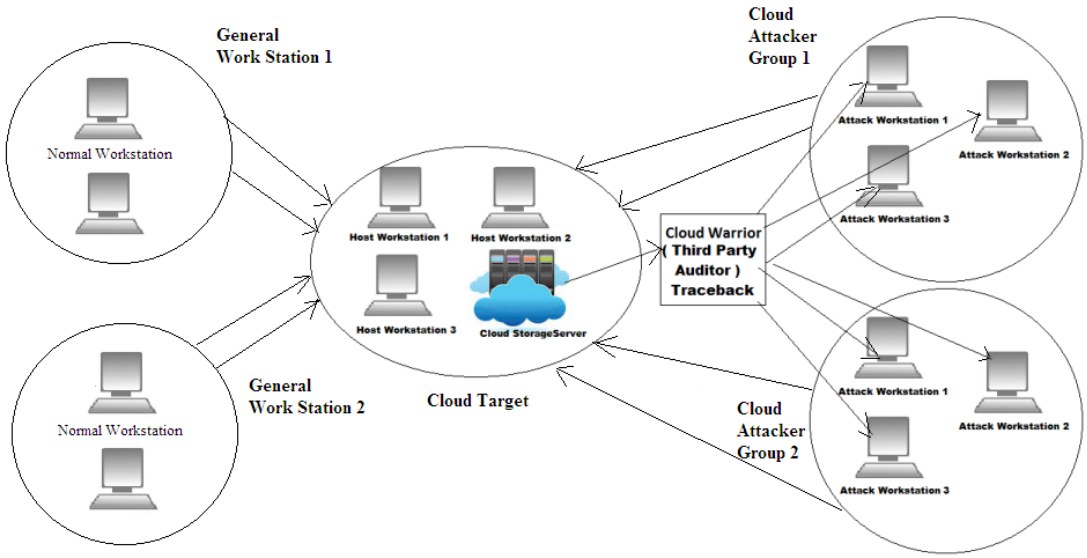


Figure 5.2: System Model of the Proposed Method

which requires very less bandwidth.

Cloud Warrior:

The Cloud Warrior (CW) is an independent, active, and trustworthy entity that filters malicious traffic from legitimate traffic and identifies the actual source of DDoS with an accurate prediction model. CW contains various essential components which are shown in Figure 5.3. These components stimulate the private cloud architecture. Description of each component is given in the following.

- **Internet**

  To initiate the cloud service, the cloud users use Internet for provisioning the resources from the Cloud Service Providers (CSP).

- **Front-end Server**

  The front-end server deploys a Cloud Fusion Unit (CFU), which is configured with the help of Cloudera CDH 5.3.0-0 [95]. This server has two Ethernet connections *eth0 and eth1*. *eth0* is connected to a public switch, which provides CSP services to the end users and *eth1* is connected to a private switch for the deployment of a virtual private cloud.

- **Virtual Private Cloud (VPC)**

  CW contains a Virtual Private Cloud (VPC) of four nodes (or VMs), which are con-

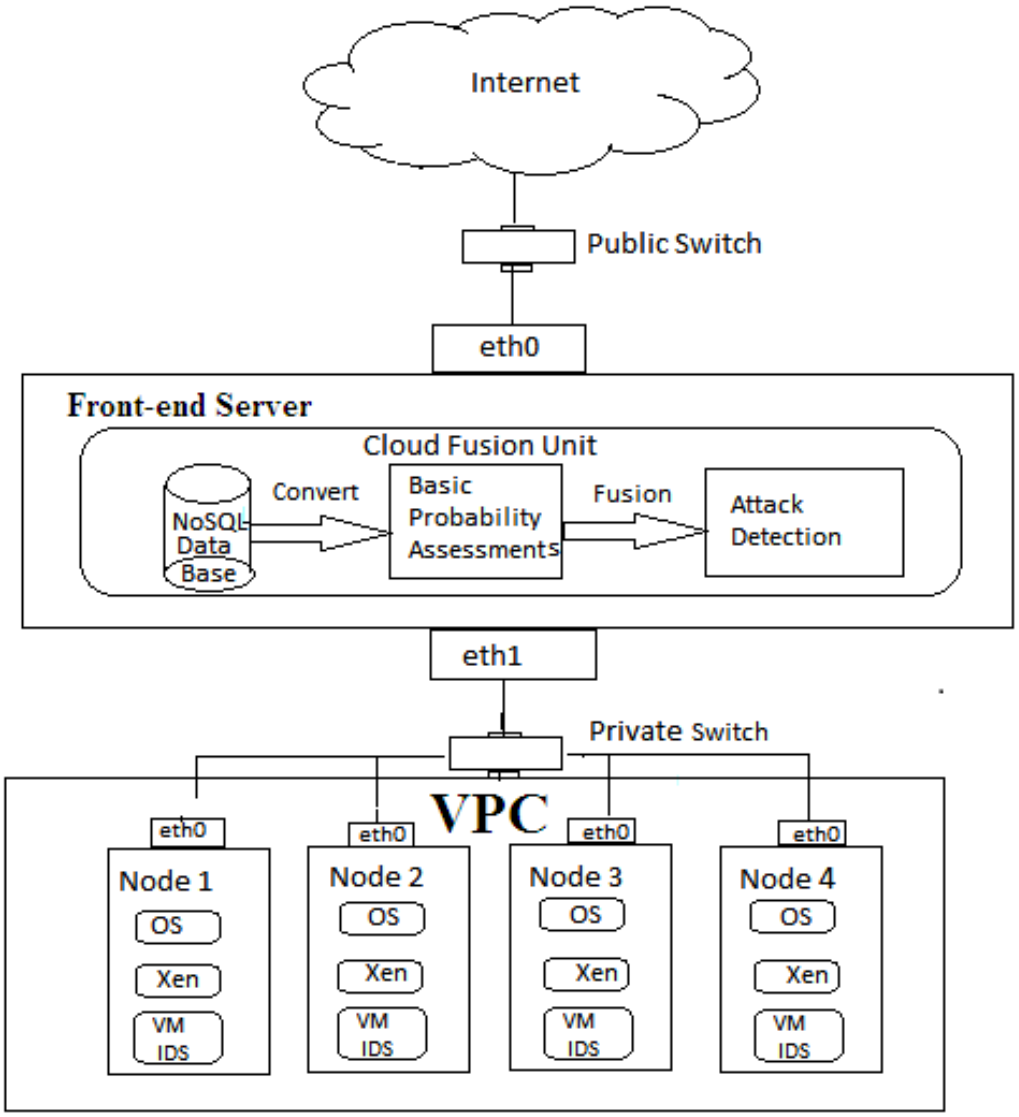


Figure 5.3: Cloud Warrior Architecture

nected with the front end server via a private switch. The Citrix Xen Server 6.2.0 [96] manages all four nodes within the "Networking Mode" because this mode perfectly delivers the advanced features of virtualization.

5.3 Proposed Technique

In cloud environment, a number of attackers may flood the network with various malicious packets which may collapse the whole structure of the systems. We describe the proposed cloud warrior based mechanism for successful detection and prevention of such DDoS at-

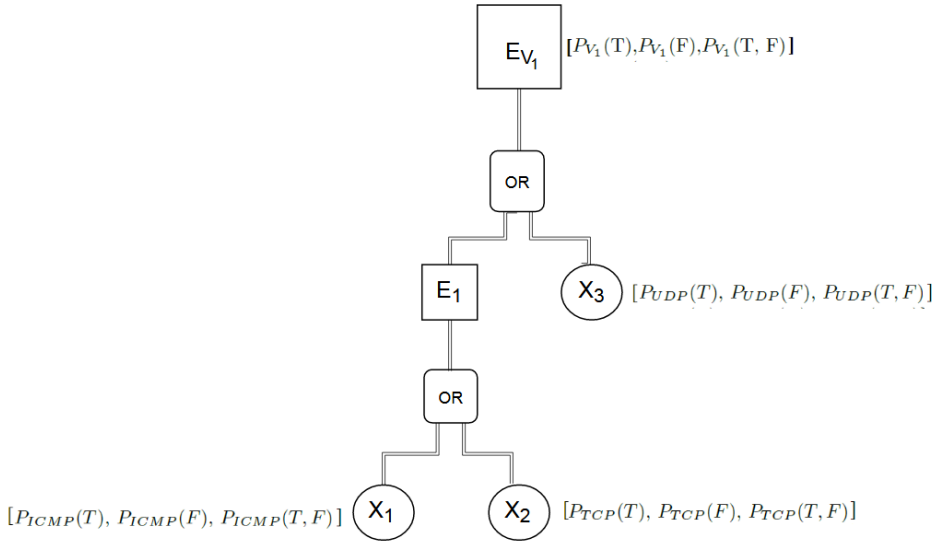


Figure 5.4: Probability Extraction Tree for VM V_1

tacks. The proposed technique is a packet trace-back approach based on TPA, which provides an interface between cloud users and Cloud Service Providers (CSPs). The Cloud Warrior works in two phases. These operation phases are described in the following.

1. **Observation:** In this phase, four virtual machine (VM) or nodes are used, which are equipped with Snort IDS tools and configured with DST and Weibull distribution. These nodes generate the attack alerts and packet floods. To generate the attack alert, VMs use a maximum threshold value of packets to avail a particular service. After attaining this value, an alert is generated for packet floods and attacks.

   Next, the front end server converts this alert into the basic probability assessment (BPA). To calculate BPA for each VM, CW uses the Probability Extraction Tree (PET), which is generated from the probability of the total packet traffic affected from the DDoS attack. This probability depends upon the packet threshold value which is decided at the VM. The BPA is stored in a NoSQL database. Figure 5.4 shows the calculation of the probabilities (i.e. P_{V_1}(T), P_{V_1}(F), P_{V_1}(T, F)) for the first VM node. Further, the procedure for conversion of BPA is described in Algorithm 5.1.

2. **Detection:** To detect the DDoS attack, we analyze the Transmission Control Protocol (TCP), Internet Control Message Protocol (ICMP) and User Datagram Protocol (UDP) packets with the help of 3-valued logic {True, False, (True, False)}. This

Algorithm 5.1 Conversion of Alerts into BPA's

Input: Alerts received from VM's .
Output: Probabilities.
$\{P_{UDP}(T), P_{UDP}(F), P_{UDP}(T,F)\}$.
$\{P_{TCP}(T), P_{TCP}(F), P_{TCP}(T,F)\}$.
$\{P_{ICMP}(T), P_{ICMP}(F), P_{ICMP}(T,F)\}$.

1: **for** each VM node **do**
2: Capture {UDP; TCP; ICMP } packets.
3: **for** each packet X $\in$ {UDP; TCP; ICMP } **do**
4: Query the alerts from the database, When a X attack occurs for the specified VM node.
5: Query the total number of possible X alerts for each VM node.
6: Query the alerts from the database when X attack is unknown.
7: Calculate the Probability(True) for X, by dividing the result obtained at step 1 with the result obtained at step 2.
8: Calculate the Probability (True, False) for X, by dividing the result obtained at step 3 with the result obtained at step 2.
9: Calculates probability (False) for X: 1- {Probability (True) + Probability(True, False)}
10: **end for**
11: Calculate the probabilities for each VM with the help of Probability Extraction Tree (PET).
12: With the help of the PET values, Belief (Bel) and Plausibility (PL) is calculated for each VM as follows :
13: Bel(V_1) = P_{V_1}(T)
14: PL(V_1) = P_{V_1}(T)+ P_{V_1}(T, F))
15: This Calculation would also be done for VM node V_2, V_3, and V_4.
16: **end for**

3-valued logic bifurcates the TCP-flood, UDP-flood and ICMP-flood attacks. After bifurcation, if their values are greater than the threshold value, then a DDoS attack from the particular port, and service is declared. Now, CW initiates the source identification from the log details by packet trace-back approach. If the probability of malicious traffic from a source IP is greater than 0.5, then CW blocks that IP.

We also observe Weibull Failure Rate function, Weibull MTTF, and Median life of the affected server to observe reliability and attack assesment for DDoS mitigation. Attack assessment is performed at the front end server, and it occupies into the CFU. It fuses the 3- valued logic and combines results of VMs for observing the impact of DDoS flood attack. These results are described in Table 5.2.

Table 5.2: Three-valued Logic Result for Four Virtual Nodes

		o_1	o_2	o_3
	$\vee$	T	F	(T, F)
v_1	T	F	T	F
v_2	T	T	T	T
v_3	F	T	F	(T, F)
v_4	(T, F)	(T, F)	T	(T, F)

As per VM set $V = \{v_1, v_2, v_3, v_4\}$ and output set $O = \{o_1, 0_2, o_3\}$, the DST 3-value logic values are

$$P(V \vee O) = \{v_1.o_2 + v_2.o_1 + v_2.o_2 + v_2.o_3 + v_3.o_1 + v_4.o_2; v_1.o_1 + v_3.o_2; v_3.o_3 + v_4.o_1 + v_4.o_3\} \tag{5.13}$$

We have the relation for DST as the [sum of all probabilities] = 1 and P(Φ) = 0 :

$$P(True) + P(False) + P(True, False) = 1 \tag{5.14}$$

So, our result can be reduced as

$$P(V \vee O) = \{v_1.o_2 + v_2 + v_3.o_1 + v_4.o_2; v_1.o_1 + v_3.o_2; v_3.o_3 + v_4.o_1 + v_4.o_3\} \tag{5.15}$$

CW can conclude from equation 5.13 that VM v_2 is the cause of DDoS attack. Thus, packet traffic from VM v_2 must be blocked. The assessment of results is maximizing the DDoS true positive alarm rates and minimizing the false positive alarm rate. It is possible by the time-invariant transition probabilities. If i and j are the independent and time invariant probability of occurring an event at time n and $n + 1$, then the equation (5.16) and (5.17) show their values.

$$Pr\{X_{n+1} = x_{n+1} | X_n = x_n\} = Pr\{X_2 = x_{n+1} | X_1 = x_n\} \tag{5.16}$$

$$Pr\{X_{n+1} = j | X_n = i\} = Pr\{X_2 = j | X_1 = i\} = P_{ij} \tag{5.17}$$

5.4 Summary

This chapter proposes an effective TPA based collaborative approach for DDoS detection and prevention. This method uses packet trace back approach for observation of all kinds of packets that reach cloud servers. TPA uses the Weibull Probability Distribution for detecting the source of the TCP flood, the UDP flood, and the ICMP flood attacks. A service model for fault analysis of each virtual machine is also provided to prevent DDoS attacks. The service model of our approach ensures the security of the CSPs. We also address the issue of IP spoofing in this work. We have analyzed the packet traffic pattern on different TCP and UDP ports and evaluated our proposed method with respect to different parameters. The results obtained with our model show that the proposed model achieves high sensitivity, specificity, and accuracy with a low false alarm rate.

Chapter 6

Experimental Results

We have performed several experiments to check the performance of the proposed authentication, data integrity verification, and DDoS attack prevention methods. In this chapter, we describe the different parameters to measure the performance of the proposed methods and present the experimental results based on these performance parameters. We also provide the comparative analysis of our methods with respect to the existing methods. In section 6.1, we present the experimental results of the proposed authentication method. Section 6.2 discusses the experimental results of the both DIV methods. Experimental results of the proposed DDoS attack prevention are presented in section 6.3. Finally, section 6.4 summarizes this chapter.

6.1 Experiment Results for Authentication Method

In this section, first we describe the experimental setup for the implementation of the proposed authentication method. Then, the performance parameters are defined for the analysis of the proposed method. Thereafter, the performance analysis with experimental results is reported.

6.1.1 Experimental Setup

We have implemented an application based on Hadoop and MapReduce framework to test our proposed authentication method. The experiments are performed on two PCs configured

with Intel Core i7-2600S 2.80 GHz and 16 GB RAM. We have configured Citrix Xen Server 6.2.0 [96] on one machine that is used for file storage. On this server, we have also configured three Virtual Machines (VMs) that access Google Cloud platform, Dropbox, and Amazon Web Service (AWS), respectively. This VM's have used three public parameters $\{P_1, P_2, P_3\}$ for secure access to the file.

The second PC configured with Cloudera CDH 5.3.0-0 [95]. It is used as a cloud user and provides access control to the stored files. The working of our scheme is divided into four phases of authentication. Figure 6.1 describes these phases.

1. **Phase 1:**

   A client program is installed on or downloaded to every endpoint (laptop, cell-phone, etc.) when the user accesses the client end. A server or gateway hosts the centralized security program, which verifies login credentials and sends updates and patches when needed. In this phase, the user contacts the Gate Keeper (GK) service in the Gateway Server (GS), where the communication with the GK (or any other service) uses Transport Layer Security to protect against eavesdropping attacks.
2. **Phase 2:**

   The GS needs to identify their users securely through authentication. After that, a user gains authorization for doing certain tasks. With the Single Sign-On Token (SSAT), a user logs in once and gains access to all systems without being prompted to log in again in each of them. The Clearance Verifier (CV) checks the validity of the token. If there is no verification in the SSAT, that service should contact the CV.
3. **Phase 3:**

   This step is a precaution against SSAT forging. If the CV reports back that the Gateway Server does not generate the SSAT, the request will be blocked. If the SSAT is examined and proved valid, the CV attaches a verification token to the SSAT.
4. **Phase 4:**

   Now, user is able to use the services.

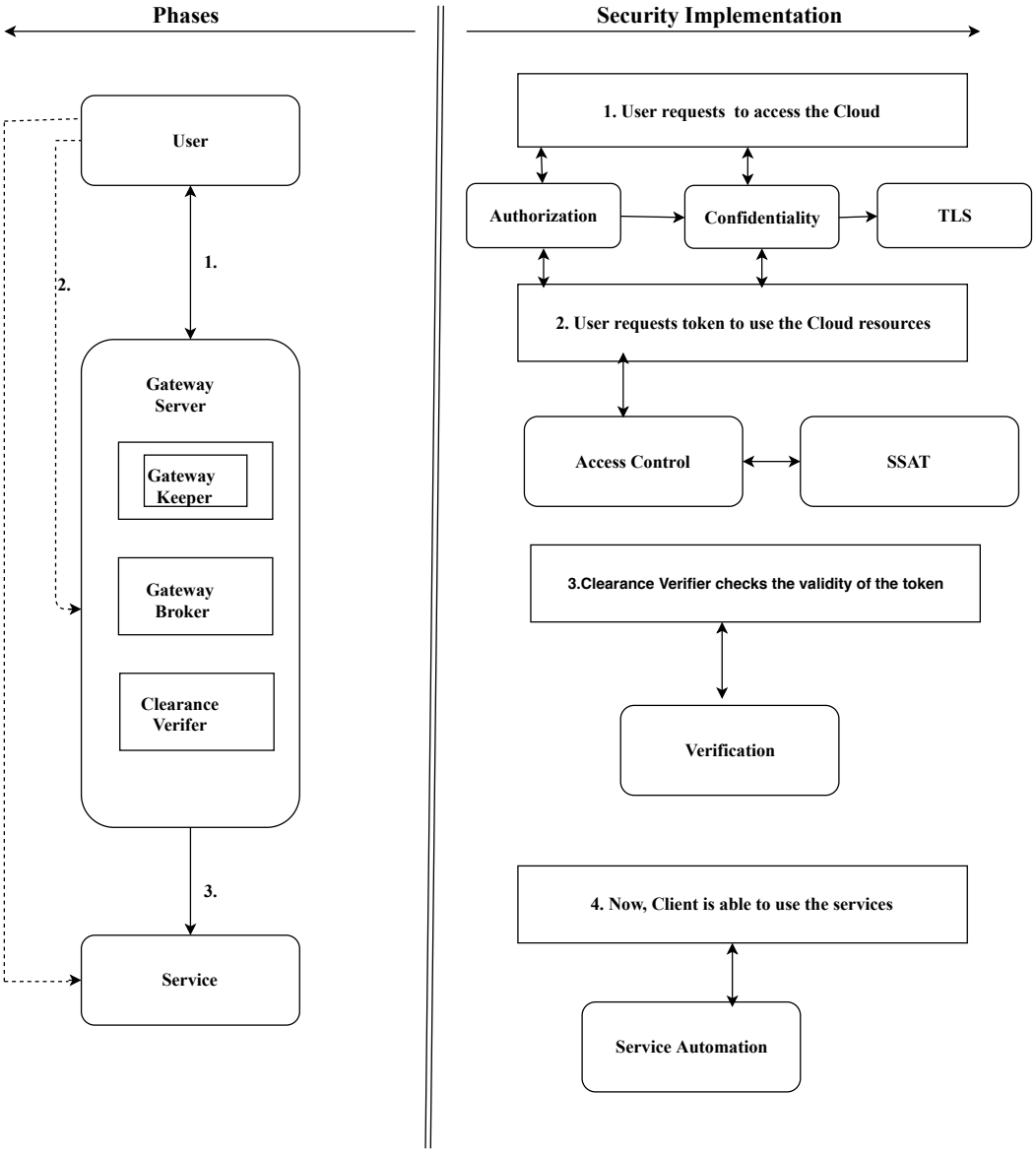


Figure 6.1: Phases of Our Authentication Method for Cloud Environment

6.1.2 Parameters

The performance parameters for authentication methods are classified into two categories (1) User Behavior Parameters, and (2) Service Level Agreement (SLA) Parameters. The description of these parameters as well as the experimental result corresponding to each parameter are given in the following paragraphs.

6.1.2.1 User Behaviour Parameters

These parameters are based on the behaviour of the cloud users and use to evaluate the ethical trust value of the user for accessing any resource from the CSP.

1. **Fake Request Rate (FRR)**

   Fake requests are the heavy load of the dummy and categorized as illogical requests that are sent to CSPs for consuming the cloud resources. Fake requests are intentionally used for implementing the DoS attack by achieving the bandwidth starvation. Thus, this parameter affects the availability of the cloud server.

   If N_{Fake} is the number of fake requests and N_{Total} is the total number of requests sent by the user in a time interval, then FRR is calculated using equation (6.1).

   $$FRR = \frac{N_{Fake}}{N_{Total}} \tag{6.1}$$

   We have evaluated FRR for different number of total requests in different time interval and assessment of FRR is reported in Table 6.1. We are able to identify on an average of 0.219 FRR at different time interval.

2. **Unauthorized Request Rate (URR)**

   Unauthorized requests are the illegal requests sent by an imposter for stealing or modifying the contents of a data file stored on Cloud server. This parameter affects the confidentiality and authorization of cloud data.

   If $N_{Unauthorized}$ is the number of unauthorized requests and N_{Total} is the total number of requests sent by the user in a time interval, then URR is represented by equa-

Table 6.1: Assessment of Fake Request Rate

$\mathbf{N_{Total}}$	$\mathbf{N_{Fake}}$	**FRR**
100	21	0.21
200	38	0.19
300	66	0.22
400	92	0.23
500	120	0.24
600	126	0.21
700	175	0.25
800	176	0.22
900	171	0.19
1000	230	0.23

Table 6.2: Assessment of Unauthorized Request Rate

$\mathbf{N_{Total}}$	$\mathbf{N_{Unauthorized}}$	**URR**
100	19	0.19
200	22	0.11
300	51	0.17
400	64	0.16
500	65	0.13
600	72	0.12
700	98	0.14
800	144	0.18
900	135	0.15
1000	165	0.17

tion (6.2).

$$URR = \frac{N_{Unauthorized}}{N_{Total}} \tag{6.2}$$

We have computed URR for different number of total requests in different time interval and assessment of URR is reported in Table 6.2. We observe an average of 0.152 URR at different time interval.

3. **Resource Affection Rate (RAR)**

   Resource Affection Rate (RAR) is the measurement of the affected resources with respect to the total number of resources accessed by a user at a time interval. This parameter affects the reliability of the cloud data.

   If $N_{Affected}$ is the number of affected resources and N_{Total} is the total number of

Table 6.3: Assessment of Resource Affection Rate

N_{Total}	$N_{Affected}$	RAR
100	9	0.09
200	20	0.10
300	24	0.08
400	44	0.11
500	75	0.15
600	72	0.12
700	91	0.13
800	88	0.11
900	108	0.12
1000	111	0.11

resources accessed in a time interval by a user, then RAR is computed by equation (6.3).

$$RAR = \frac{N_{Affected}}{N_{Total}} \tag{6.3}$$

Assessment of RAR for different values is given in Table 6.3. From Table 6.3, we can see that RAR varies from 0.08 to 0.15 for different number of total request.

6.1.2.2 SLA Parameters

SLA parameters evaluate the successful agreement between the cloud user and CSPs at the time of service agreement. It also judges the requirements of cloud user that are fulfilled by CSPs. These parameters observe the ability of CSP for providing services to the user.

1. **Turn Around Efficiency (TAE)**

   We assume that $R_1, R_2.....R_k$ are the cloud resources available for end users. Turn Around Efficiency (TAE) is the ratio of the total number of submitted job by a user for the resource R_k and the number of the successfully completed job using R_k. If $N_k(R_k)$ is the total number of jobs submitted for resource R_k and $S_k(R_k)$ is the actual number of jobs successfully completed using resource R_k, then Turn Around Efficiency (TAE) of resource R_k is calculated using equation (6.4).

$$TAE(R_k) = \frac{S_k(R_k)}{N_k(R_k)} \tag{6.4}$$

Table 6.4: Assessment of Turn Around Efficiency

$\mathbf{N_k(R_k)}$	$\mathbf{S_k(R_k)}$	$\mathbf{TAE(R_k)}$
40	33	0.83
60	54	0.90
64	58	0.91
70	60	0.86
75	67	0.89
80	75	0.94
86	79	0.92
90	85	0.94
95	90	0.95
100	95	0.95

To measure TAE, we consider different sets of random jobs for particular resources and examine the number of successfully completed jobs from the submitted jobs. Results for a particular resource are reported in Table 6.4. We observe that TAE varies between 0.83 to 0.95.

2. **Resource Availability (RA)**

   If the cloud resources are affected by malicious attackers then they are not available for executing the jobs. $N_k(R_k)$ denotes the number of jobs assigned to each R_k in a duration of time T. If $A_k(R_k)$ represents the number of jobs accepted by R_k, then RA of R_k is represented by equation (6.5).

$$RA(R_k) = \frac{A_k(R_k)}{N_k(R_k)} \tag{6.5}$$

   To evaluate RA, we utilize the earlier sets of random jobs and check whether a particular resource is allocated for those jobs or not. RAs for different sets of jobs are given in Table 6.5. We achieve an average of 0.945 RA.

3. **Successful Transaction Rate (STR)**

   It defines the rate of successfully completed jobs assigned to a resource R_k. We assume that $A_k(R_k)$ is the number of jobs accepted for execution by R_k. $S_k(R_k)$ is the number of jobs successfully completed by the R_k over the period T. The STR

Table 6.5: Assessment of Resource Availability

$\mathbf{N_k(R_k)}$	$\mathbf{A_k(R_k)}$	$\mathbf{RA(R_k)}$
40	36	0.90
60	56	0.93
64	60	0.94
70	64	0.91
75	70	0.93
80	78	0.98
86	80	0.93
90	87	0.97
95	93	0.98
100	98	0.98

Table 6.6: Assessment of Successful Transaction Rate

$\mathbf{A_k(R_k)}$	$\mathbf{S_k(R_k)}$	$\mathbf{STR(R_k)}$
36	33	0.92
56	54	0.96
60	58	0.97
64	60	0.94
70	67	0.96
78	75	0.96
80	79	0.99
87	85	0.98
93	90	0.97
98	95	0.97

for R_k is calculated using equation (6.6).

$$STR(R_k) = \frac{S_k(R_k)}{A_k(R_k)} \tag{6.6}$$

Here, we also consider the same sets of random jobs assigned to a particular resource and find out the successfully completed job for that resource. The results are reported in Table 6.6. An average of 0.96 STR is achieved in this experiment.

4. **Integrity Preservation (IP)**

   It defines Integrity Preservation (IP) of jobs assigned to a resource R_k. We assume that $C_k(R_k)$ is the number of jobs preserved the integrity of a total number of jobs $D_k(R_k)$ assigned to a resource R_k over a time period T. The IP for R_k is evaluated

Table 6.7: Assessment of Integrity Preservation

$\mathbf{C_k(R_k)}$	$\mathbf{D_k(R_k)}$	$\mathbf{IP(R_k)}$
26	27	0.96
33	33	1.00
45	45	1.00
46	47	0.98
53	53	1.00
57	57	1.00
61	62	0.98
65	65	1.00
76	77	0.99
85	85	1.00

by equation (6.7).

$$IP(R_k) = \frac{C_k(R_k)}{D_k(R_k)} \tag{6.7}$$

In this experiment, we consider different sets of jobs for a particular resource and check whether successfully completed jobs preserved the integrity or not. Assessment of Integrity Preservation for different sets of jobs is given in Table 6.7. We achieve an average of 0.99 IP for our approach.

6.1.3 Performance Analysis

After evaluating the individual parameter, we compute the overall user and CSP performances of the proposed authentication scheme.

- **User Performance**

  Overall user performance is computed as a weighted sum of user's behaviour parameters. W_1, W_2, and W_3 are the weighted values of FRR, URR, and RAR, respectively. Then the User Performance (UP) at a given time interval is evaluated by equation (6.8).

$$UP = [1 - ((W_1 * FRR) + (W_2 * URR) + (W_3 * RAR))] * 100 \tag{6.8}$$

We have empirically chosen value for W_1, W_2, and W_3 as $W_1 = 0.1$, $W_2 = 0.2$ and

Table 6.8: Assessment of User Performance

FRR	URR	RAR	UP
0.21	0.19	0.09	91.40
0.19	0.11	0.10	92.90
0.22	0.17	0.08	92.00
0.23	0.16	0.11	91.20
0.24	0.13	0.15	90.50
0.21	0.12	0.12	91.90
0.25	0.14	0.13	90.80
0.22	0.18	0.11	90.90
0.19	0.15	0.12	91.50
0.23	0.17	0.11	91.00

$W_3 = 0.3$. The assessment of UP at a given time interval is reported in Table 6.8. We achieve an average of 91.41% user performance at a given time interval for the proposed system.

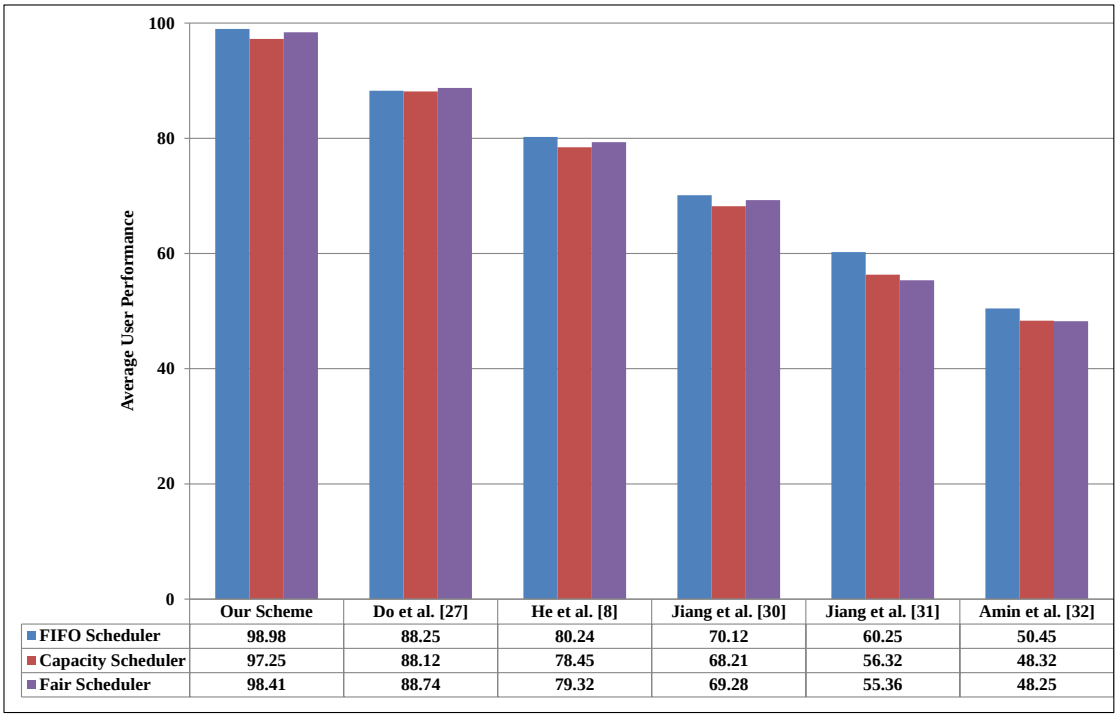
We have also computed the Average User Performance (AUP) for a given set of time intervals. To evaluate the Average User Performance (AUP), we use the UP values for the given time intervals. Let i = 1, 2, ..., t_n be the time intervals for which AUP is to be measured. The Average User Performance (AUP) is calculated using equation (6.9).

$$AUP = \frac{\sum_{i=1}^{t_n} UP_i}{t_n} \tag{6.9}$$

We have evaluated AUP with three schedulers, namely FIFO, Capacity, and Fair to compare the average user performance with some existing techniques. The comparative analysis of AUP with three different schedulers for different methods is shown in Fig. 6.2. It can be observed that the proposed scheme gives 93.97%, 91.25%, and 94.41% AUPs for FIFO, Capacity, and Fair schedulers, respectively. These are the highest AUPs among all existing methods.

- **CSP Performance**

  Overall CSP performance for a resource is evaluated based on the individual SLA parameters. This is computed as a weighted sum of TAE, RA, STR, and IP parameters as given in equation (6.10). In equation (6.10), Z_1, Z_2, Z_3, and Z_4 are the weighted



	Our Scheme	Do et al. [27]	He et al. [8]	Jiang et al. [30]	Jiang et al. [31]	Amin et al. [32]
FIFO Scheduler	98.98	88.25	80.24	70.12	60.25	50.45
Capacity Scheduler	97.25	88.12	78.45	68.21	56.32	48.32
Fair Scheduler	98.41	88.74	79.32	69.28	55.36	48.25

Figure 6.2: Comparative Analysis of Average User Performance

values of TAE, RA, STR, and IP, respectively.

$$CSPP = [(Z_1 * TAE) + (Z_2 * RA) + (Z_3 * STR) + (Z_4 * IP)] * 100 \quad (6.10)$$

We have empirically chosen the values of Z_1, Z_2, Z_3, and Z_4 as follows: $Z_1 = 0.1$, $Z_2 = 0.2$, $Z_3 = 0.3$, and $Z_4 = 0.4$. The assessment of $CSPP$ for a given resource is reported in Table 6.9. We achieve an average of 96.49% of CSP performance.

6.2 Experimental Results of Data Integrity Verification Methods

In this section, first we present the experimental setup for the DIV methods. Then, the performance parameters are defined for the analysis of the proposed method. Thereafter, the performance and security analysis are reported. A comparative study with the existing methods is also done in this section.

Table 6.9: Assessment of CSP Performance

$\mathbf{TAE(R_k)}$	$\mathbf{RA(R_k)}$	$\mathbf{STR(R_k)}$	$\mathbf{IP(R_k)}$	**CSPP**
0.83	0.90	0.92	0.96	92.30
0.90	0.93	0.96	1.00	96.40
0.91	0.94	0.97	1.00	97.00
0.86	0.91	0.94	0.98	94.20
0.89	0.93	0.96	1.00	96.30
0.94	0.98	0.96	1.00	97.80
0.92	0.93	0.99	0.98	96.70
0.94	0.97	0.98	1.00	98.20
0.95	0.98	0.97	0.99	97.80
0.95	0.98	0.97	1.00	98.20

6.2.1 Experimental Setup

To measure the performance of our DIV approaches, we have executed a series of experiments on two PCs configured with Intel Core i7-2600S 2.80 GHz and 16 GB RAM. We establish a cloud setup with Cloudera CDH 5.3.0-0 [95]. To gain the access of file storage server, we have used Citrix Xen Server 6.2.0 [96]. We have used 1 TB internal hard-disk for data storage. One PC is used as a TPA group that audits the stored files on behalf of cloud users.

6.2.2 Parameters

From the performance point of view, we focus on how frequently and efficiently a user can verify his stored data from CSP without retrieving it. In our scheme, the total number of verification and the number of challenge blocks required for each verification can be decided according to user's requirement. If data are not stored for a long time, user can set a small number of verification and challenge blocks which reduce the overload of TPA also. We have consider the following parameters to measure the efficacy of the proposed DIV approaches.

- **Probability of Server Misbehaviour Detection:** If storage server deletes r file blocks and the TPA group detects the server misbehaviour after a challenge with c file blocks, then we compute P_X, the probability that at least one of the blocks picked by TPA

group matches one of the blocks deleted by the server with equation (6.11).

$$P_X = P\{X \geqslant 1\} = 1 - P\{X = 0\} = 1 - \{\frac{n-r}{n}.\frac{n-1-r}{n-1}.\frac{n-2-r}{n-2}....\frac{n-c+1-r}{n-c+1}\} \tag{6.11}$$

P_X indicates the probability of server misbehaviour detection which depends on the total number of file blocks n, deleted file blocks r, and challenged file blocks c. X is a discrete random variable which defines the number of blocks chosen by TPA group that matches the blocks deleted by the server.

- **Accuracy:** Accuracy or Classification Rate (Acc_R) is the percentage of truly verified blocks ($T_P + T_N$) to the total number of blocks actually given for verification ($T_P + F_P + T_N + F_P$).

$$Acc = \frac{T_P + T_N}{T_P + F_P + T_N + F_P} \tag{6.12}$$

where, T_P = True Positives (Correctly verified file blocks for which data has not been modified and verification result also claims that data has not been modified), F_P = False Positives (Mistakenly verified file blocks for which data has been modified and verification result claims that data has not been modified), T_N = True Negatives (Correctly verified file blocks for which data has been modified and verification result also claims data has been modified) and F_N = False Negatives (Mistakenly verified file blocks for which data has not been modified and verification result claims that data has been modified).

- **Detection Rate or Sensitivity:** Detection rate or sensitivity (D_R) is the measurement of the fraction of correctly verified file blocks for which data has not been modified. It is computed as the percentage of true positives over the sum of true positives and false negatives as given in equation (6.13).

$$D_R = \frac{T_P}{T_P + F_N} \times 100 \tag{6.13}$$

where T_P represents True Positives, and F_N represents False Negatives.

- **Availability:** Availability measures the availability of verification information from active namenodes and reverse namenodes while verification is performed. It is calcu-

lated using equation (6.14). In equation (6.14), N_{nodes} denotes the number of times verification information available from namenodes and $N_{verification}$ is the total number of verification to be performed.

$$Ava = \frac{N_{node}}{N_{verification}} \times 100 \tag{6.14}$$

- **Performance:** Performance of the proposed DIV methods is measured as a percentage of successfully verified file blocks. It is calculated using equation (6.15).

$$Performance = \frac{Number\ of\ verified\ blocks}{Total\ number\ of\ blocks\ submitted\ for\ verification} \times 100 \tag{6.15}$$

- **Verification Delay:** Verification delay is the time taken to verify file blocks. In our experiment, we computed verification delay (VD) using equation (6.16). In equation (6.16), T_s is the time-stamp when verification process of the file blocks starts and T_c is the time-stamp when verification completes.

$$VD = T_c - T_s \tag{6.16}$$

- **Verification Overhead:** Verification overhead is the time taken to complete the verification of file blocks from the time of submission. In our experiment, we computed verification overhead (VO) using equation (6.17). In equation (6.17), T_i is the time-stamp when verification requests of file blocks are sent and T_c is the time-stamp when verification process completes. This also includes the verification delay.

$$VO = T_i - T_s \tag{6.17}$$

6.2.3 Performance Analysis of DIV Methods

We have analyzed the performance of the DIV methods with respect to the parameters defined above. We have also done a comparative study of some important existing work with respect to these parameters. To evaluate the performance of different parameters, we use different experimental configuration. The description of experimental configuration and evalu-

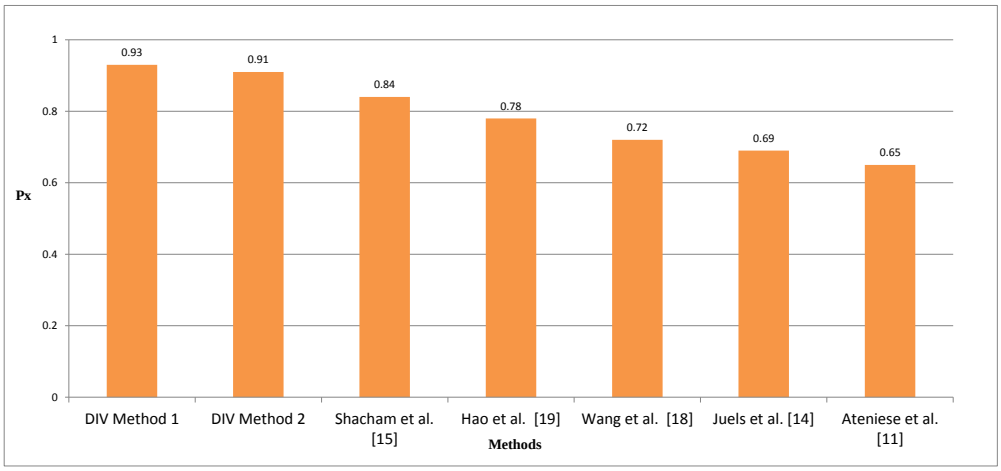


Figure 6.3: Comparative Analysis of Probability of Server Misbehaviour Detection for DIV Methods

ated results corresponding to each parameters are given below.

- **Probability of Server Misbehaviour Detection:** To compute probability of server misbehaviour detection, we use a file of size 1 TB. The file is divided into n blocks ($n = 16384$) and stored into different CSPs. We consider the size of the block as 64 MB. To perform the data integrity check we use 164 challenge blocks (c). To calculate P_x, we check how many random blocks are required to perform the verification of the challenged blocks without any server misbehaviour. The results of probability of server misbehaviour detection (P_x) for both DIV methods are reported in Fig. 6.3. We achieve P_x of 0.93 and 0.95 for our proposed DIV method 1 and DIV method 2, respectively. From Fig. 6.3, we can observe that both the proposed methods give higher P_x values than the existing methods. We achieve better results because algebraic signature and PHC properties provide unique tag value for verification which requires less number of operations at the time of verification as compared to the existing methods.
- **Accuracy and Sensitivity:** To calculate the accuracy and sensitivity, we deliberately modify the content of some stored file blocks. We have chosen 100, 200, 300, 400, and 500 random file blocks for verification and examine the verification results for true positives, false positives, true negatives, and false negatives. The evaluated results for DIV method 1 and DIV method 2 are reported in Table 6.10 and Table 6.10, respectively. We achieve average accuracy of 95% and 92.76% for different set of

Table 6.10: Accuracy Assessment of Our DIV Method 1

Total Sample	True Positives	True Negatives	False Positives	False Negatives	Accuracy (%)	Sensitivity (%)
100	65	28	2	5	93.00	92.86
200	128	63	4	5	95.50	96.24
300	198	87	8	7	95.00	96.59
400	287	93	13	7	95.00	97.62
500	389	93	10	8	96.4	97.98

Table 6.11: Parameters Assessment of Our DIV Method 2

Total Sample	True Positives	True Negatives	False Positives	False Negatives	Accuracy %	Sensitivity (%)
100	67	23	5	6	90.00	91.78
200	131	51	9	9	91.00	93.57
300	197	81	10	12	92.67	94.26
400	291	92	12	5	95.75	98.31
500	383	89	16	12	94.4	96.96

verification blocks in DIV method 1 and DIV method 2, respectively. We also achieve average sensitivity of 96.26% and 94.86% in DIV method 1 and DIV method 2, respectively.

Further, we have compared the average accuracy of the proposed DIV methods with some important existing works. The comparative results are presented in Fig. 6.4. From Fig. 6.4, we can observe that both the proposed methods outperform the existing methods.

- **Availability:** We use active namenodes and reserve namenodes for the verification of data integrity. Active namenodes have the information of data blocks where they have been stored. When the CSP is not able to access the active namenodes, then unavailability of data blocks is occurred. Due to unavailability of active namenodes, reserve namenodes are assigned for the verification process. To check the availability of namenodes, we consider 820 queried blocks to verify 1 TB file. In DIV method 1, active namenodes are available on an average 93.12 % at the time of verification process, while reserve namenodes are available on an average of 97.78%. Similarly, in DIV method 2, active namenodes and reverse namenodes are available on an average of 92.45% and 96.89%, respectively. The availability results of the proposed methods

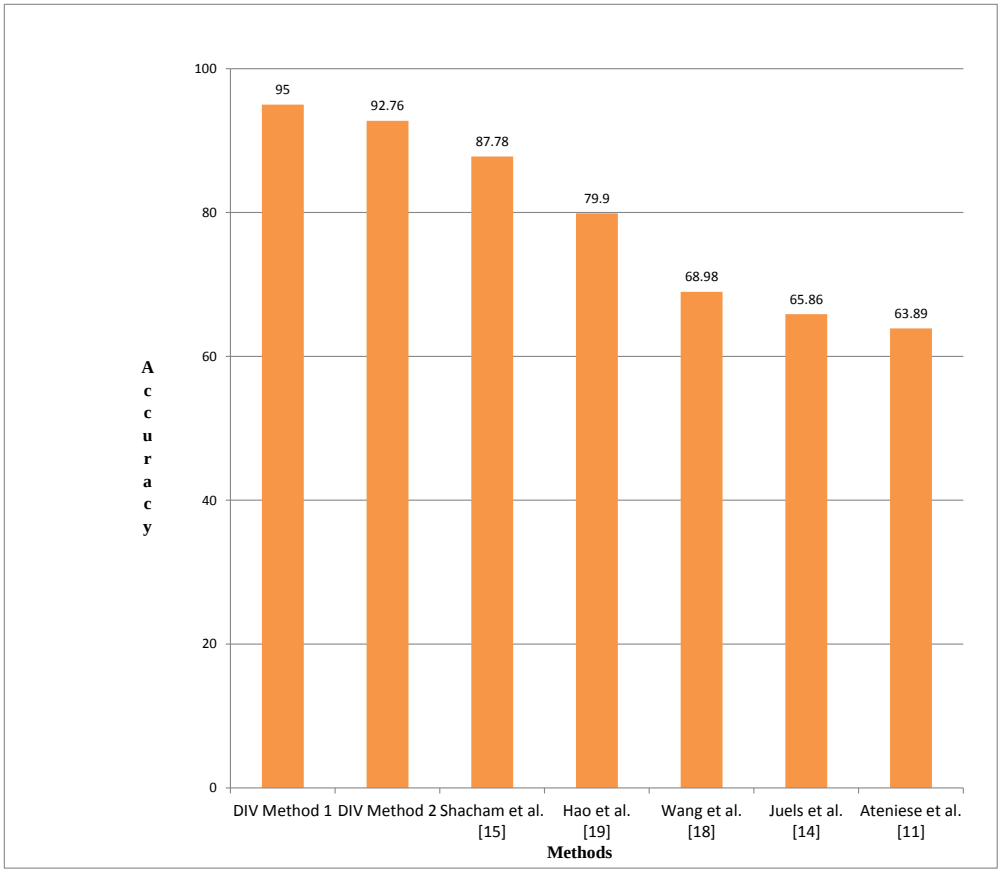
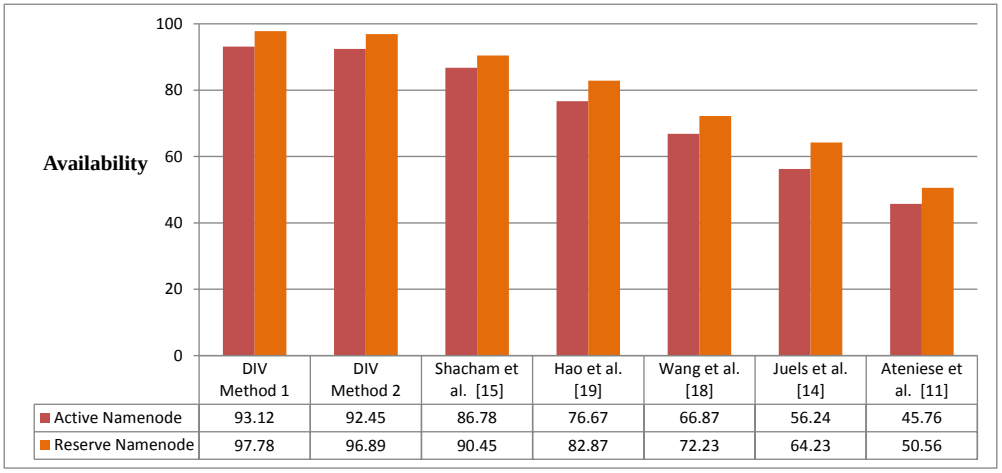


Figure 6.4: Comparative Results of Accuracy Parameter for DIV Methods

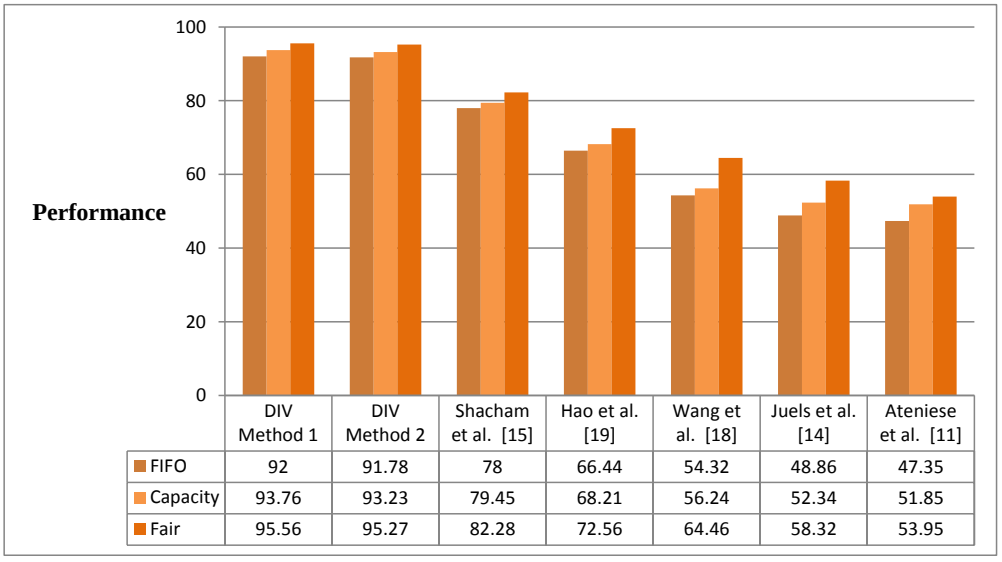
along with the comparative analysis of different methods are shown in Fig. 6.5. It is evident from Fig. 6.5 that active namenodes and reverse namenodes are available maximum number of times than the existing methods.

- **Performance:** We use three different scheduler FIFO, Capacity, and Fair for verification process to calculate the performance. These schedulers are used for verification scheduling of data blocks in active and reserve namenodes. The performances of DIV method 1 with FIFO, Capacity, and Fair schedulers are 92%, 93.76%, and 95.56%, respectively. Similarly, the performances of DIV method 2 are 91.78%, 93.23%, and 95.27% for FIFO, Capacity, and Fair schedulers, respectively. From the comparative performances analysis of different methods as given in Fig. 6.6, we observed that both the proposed methods outperform the existing methods with all type of schedulers.
- **Verification Delay:** We have considered different set of queried file blocks of size 100 to 1000 to compute verification delay. We measure the average verification delay for each set of queried blocks, and the results for proposed DIV methods are reported



	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
Active Namenode	93.12	92.45	86.78	76.67	66.87	56.24	45.76
Reserve Namenode	97.78	96.89	90.45	82.87	72.23	64.23	50.56

Figure 6.5: Comparative Results of Availability Parameter for DIV Methods



	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
FIFO	92	91.78	78	66.44	54.32	48.86	47.35
Capacity	93.76	93.23	79.45	68.21	56.24	52.34	51.85
Fair	95.56	95.27	82.28	72.56	64.46	58.32	53.95

Figure 6.6: Performance Comparison of DIV Methods

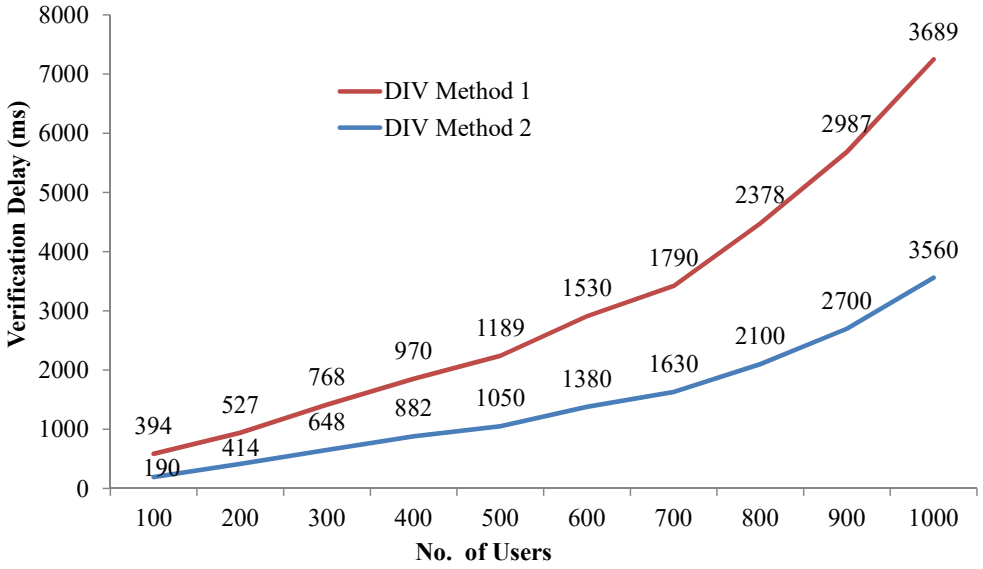


Figure 6.7: Verification Delay for the Proposed DIV Methods

in Figure 6.7. We observe that DIV method 2 completes verification tasks with less verification delay than the DIV method 1.

- **Verification Overhead:** We have evaluated the verification overhead of the same set of queried file blocks which are used to compute the verification delay. Verification overhead is the time taken to complete and verify file blocks at the time of submission. To calculate the verification overhead, we need two time stamp values, first when verification requests of file blocks are sent, and second, when verification process completes. The difference of these values is called as verification overhead. Comparative analysis of two proposed DIV methods is given in Figure 6.8 and Fig. 6.9 for 95% , 99% , and all blocks, respectively.

Further, we have analyzed the ratio of queried blocks (challenged blocks) and the total number of verified blocks by varying the different parameters.

First, we vary the size of the stored files from 128 GB to 1 TB. These files are divided into a number of blocks of size 64 MB and stored into different CSPs. We measure the ratio between queried blocks and the total number of blocks for our both DIV methods. We have also measured the same for some existing DIV approaches. Figure 6.10 shows the comparative results for different file sizes. We achieve the ratio of 23.87% and 22.75% for our proposed DIV method 1 and DIV method 2, respectively. We can observe that this ratio is very less than the existing methods.

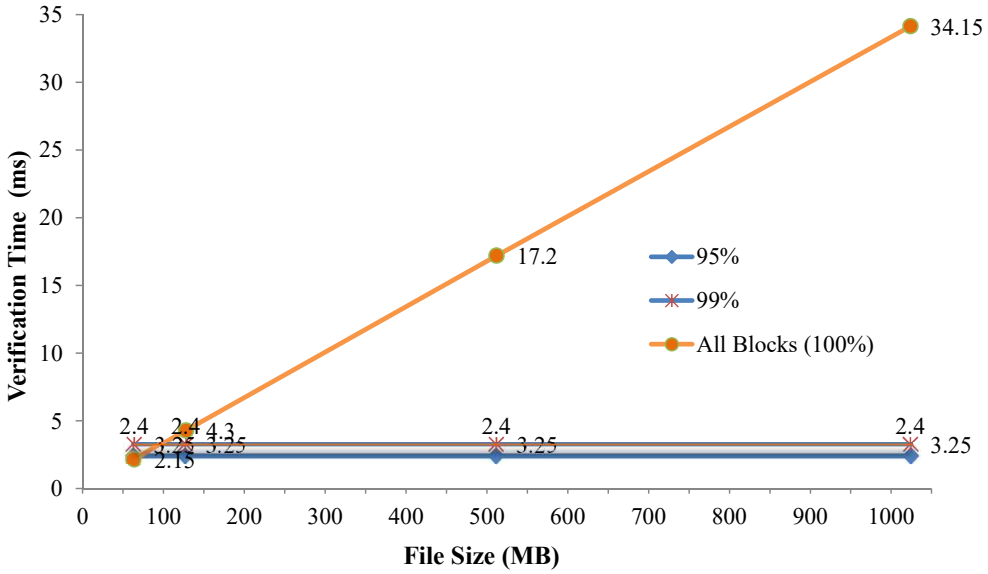


Figure 6.8: Verification Overhead for the Proposed DIV Method 1

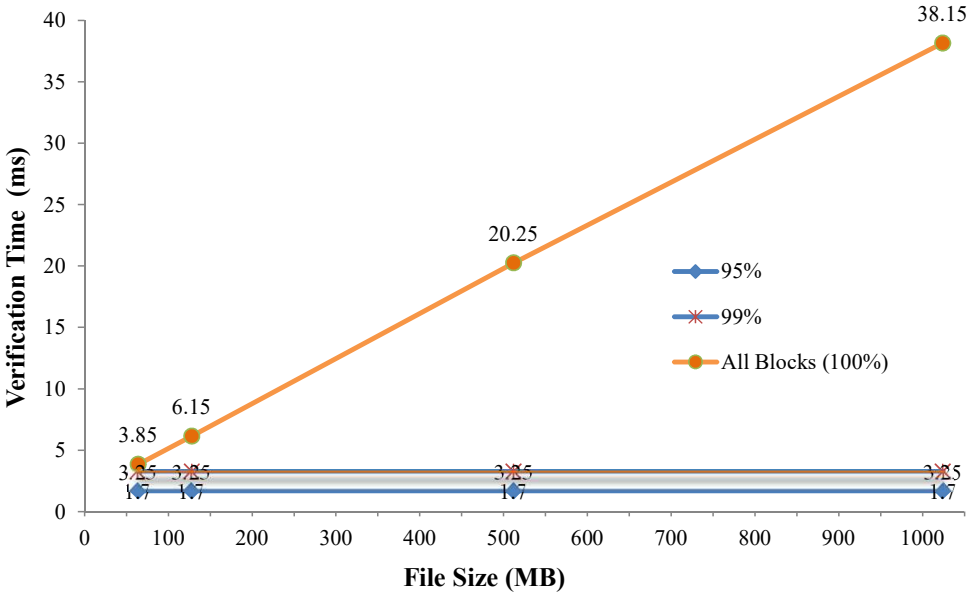
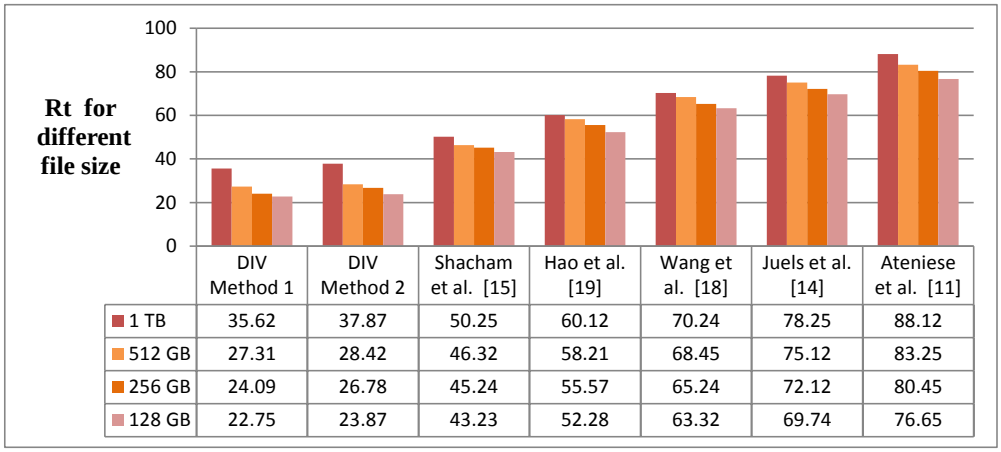
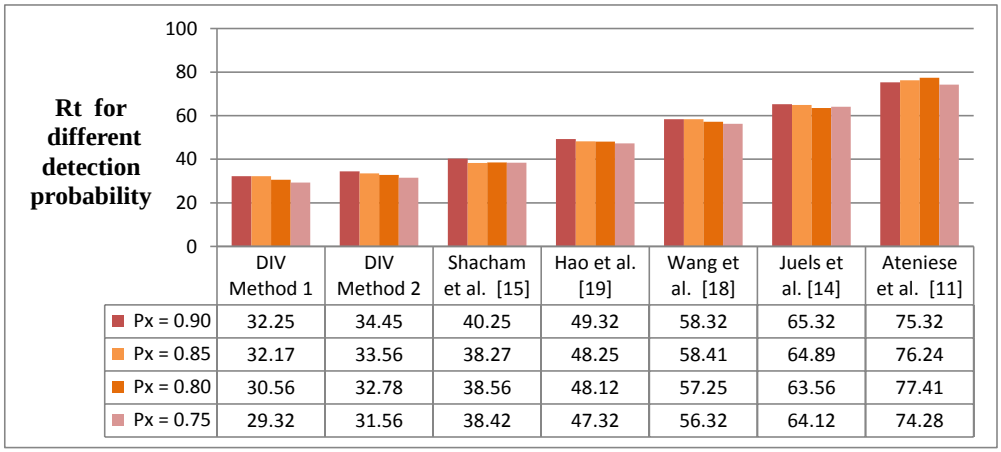


Figure 6.9: Verification Overhead for the Proposed DIV Method 2



	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
1 TB	35.62	37.87	50.25	60.12	70.24	78.25	88.12
512 GB	27.31	28.42	46.32	58.21	68.45	75.12	83.25
256 GB	24.09	26.78	45.24	55.57	65.24	72.12	80.45
128 GB	22.75	23.87	43.23	52.28	63.32	69.74	76.65

Figure 6.10: Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different File Size

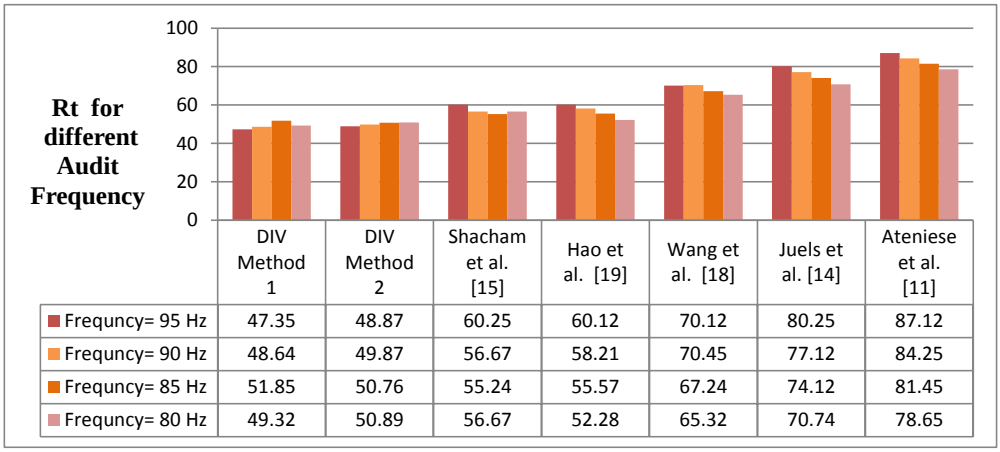


	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
Px = 0.90	32.25	34.45	40.25	49.32	58.32	65.32	75.32
Px = 0.85	32.17	33.56	38.27	48.25	58.41	64.89	76.24
Px = 0.80	30.56	32.78	38.56	48.12	57.25	63.56	77.41
Px = 0.75	29.32	31.56	38.42	47.32	56.32	64.12	74.28

Figure 6.11: Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Detection Probabilities

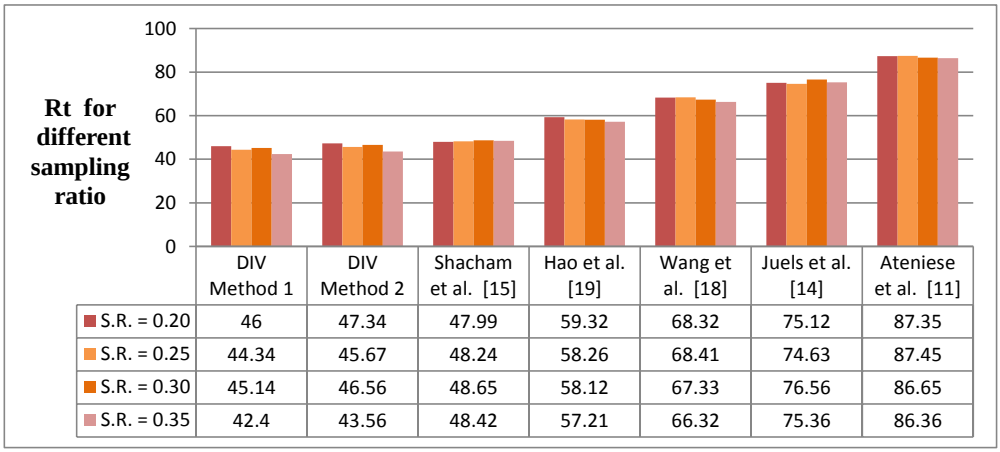
Next, we vary the probability of server misbehaviour detection from 0.75 to 0.90. We measure the ratio between queried blocks and the total number of blocks for our both DIV methods and also compare with some existing DIV approaches. Figure 6.11 shows the comparative study with different probability of server misbehaviour. We achieve the ratio of 31.56% and 29.32% for our proposed DIV method 1 and DIV method 2, respectively. We can observe that this ratio is very less than the existing methods.

We change audit frequency from 80Hz to 95Hz. We measure the ratio between queried blocks and the total number of blocks for our both DIV methods and also compare with some existing DIV approaches. Figure 6.12 shows the comparative study with different probability



	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
Frequncy= 95 Hz	47.35	48.87	60.25	60.12	70.12	80.25	87.12
Frequncy= 90 Hz	48.64	49.87	56.67	58.21	70.45	77.12	84.25
Frequncy= 85 Hz	51.85	50.76	55.24	55.57	67.24	74.12	81.45
Frequncy= 80 Hz	49.32	50.89	56.67	52.28	65.32	70.74	78.65

Figure 6.12: Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Audit Frequency



	DIV Method 1	DIV Method 2	Shacham et al. [15]	Hao et al. [19]	Wang et al. [18]	Juels et al. [14]	Ateniese et al. [11]
S.R. = 0.20	46	47.34	47.99	59.32	68.32	75.12	87.35
S.R. = 0.25	44.34	45.67	48.24	58.26	68.41	74.63	87.45
S.R. = 0.30	45.14	46.56	48.65	58.12	67.33	76.56	86.65
S.R. = 0.35	42.4	43.56	48.42	57.21	66.32	75.36	86.36

Figure 6.13: Comparative Results of the Ratio of Queried Blocks and the Total Number of Verified Blocks for Different Sampling Ratio

of server misbehaviour. We achieve the ratio of 50.89% and 49.32% for our proposed DIV method 1 and DIV method 2, respectively. We can observe that this ratio is less than the existing methods.

Finally, we vary sampling ratio from 0.35 to 0.20 and measure the ratio between queried blocks and the total number of blocks for our both DIV methods and also compare with some existing DIV approaches. Figure 6.13 shows the comparative study with different probability of server misbehaviour. We achieve the ratio of 42.4% and 43.56% for our proposed DIV method 1 and DIV method 2, respectively. We can observe that this ratio is less than the existing methods.

6.3 Experiment Results of DDoS Attack Prevention Method

We have executed several experiments to measure the performance of the proposed DDoS attack prevention method. In this section, first, we define different parameters to evaluate the efficiency of the method. Next, we set a test plan to execute our proposed scheme. After that, we introduce a hybrid test model of Agent Handler (AH) model [101], Internet Relay Chat (IRC) Mmodel [101] and Web-based Model [101]. Finally, we detect the prominent DDoS attack and report the result for Cloud Environment.

6.3.1 Parameters

1. **Detection Rate (D_R) or Sensitivity**

   Detection Rate or Sensitivity is the measurement fraction of attack pattern that is correctly detected or selected attack packets. It is the ratio of true positives to the sum of true positives and false negatives.

$$(D_R) = Sensitivity = \frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{T_P}{T_P + F_N} \quad (6.18)$$

   Where T_P = True Positives (Correctly selected) and F_N = False Negatives (Mistakenly Rejected).

2. **Specificity**

   Specificity is the measurement fraction of attack pattern that have correctly rejected. It is the ratio of true negatives to the sum of true negatives and false positives.

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} = \frac{T_N}{T_N + F_P} \quad (6.19)$$

   Where T_N = True Negatives (Correctly Rejected) and F_P = False Positives (Mistak-

enly Selected).

3. **False Alarm Rate**

   False Alarm Rate is the measurement fraction of attack pattern that is mistakenly selected. It is the ratio of false positives to the sum of true negatives and false positives.

$$False\ Alarm\ Rate = (1 - Specificity) = \frac{False\ Positives}{True\ Negatives + False\ Positives} = \frac{F_P}{T_N + F_P} \tag{6.20}$$

   Where T_N = True Negatives (Correctly Rejected) and F_P = False Positives (Mistakenly Selected).

4. **Accuracy**

   Accuracy or Classification Rate (C_R) is the ratio of true classified events($T_P + T_N$) to the total number of actually occurred events ($T_P + F_P + T_N + F_P$).

$$Accuracy = Classification\ Rate(C_R) = \frac{T_P + T_N}{T_P + F_P + T_N + F_P} \times 100 \tag{6.21}$$

   Where, T_P = True Positives (Correctly selected), F_P = False Positives (Mistakenly Selected), T_N = True Negatives (Correctly Rejected) and F_N = False Negatives (Mistakenly Rejected).

6.3.2 Test Plan

We have designed to test our scheme with the following plan and configuration.

1. A web-site is used to test, which is hosted on Cloud server. For this purpose, we used a commercial web-site “http://www.healthcont.com/,” which is hosted on Amazon AWS.
2. Backtrack tools are used to perform the attacks on this web-site.
3. One machine is configured as a TPA. It works as “ Cloud Warrior" private cloud that is implemented using Cloudera CDH 5.3.0-0 [95].
4. TPA collects the log of servers. In this log, we analyze and monitor the packet traffic

for various services. There are lots of TCP and UDP ports available for different cloud services. We choose TCP port no. 445, 6886, 16888, 11740, 23791,23830, 23980, and 23981 because we found the huge amount of traffic on these port. Similarly, We also analyzed the UDP packet traffic on UDP port no. 53, 1863, 13347, 20284, 20339, 23981, 31327, 40971, and 42847.

5. We also include the comparison based on different performance and security parameters for this proposed technique with other methods.

6.3.3 Test Model

Our test model is based on the hybrid test model of AH, IRC, and web-based model, which observed the Botnet and Darknet domain of IP addresses in Cloud environment. A bot is a software application that executes and runs the automated tasks over the Internet. Bots perform tasks faster from the human being in both the ways; simple and structurally repetitive. A botnet is composed of a group of Bots, which are the programs operating in an automated way, also commonly refer to the hosts that compromised by any means of malware.

Darknet is a set area of unused IP addresses. In Darknet, there is no application service like web and mail service. Thus, the user is not able to transfer packets to the Darknet except some misconfigured packets. If the discovery methods is based on round robin or random schedule, then few packets are transferred towards the Darknet. It is possible in scan attack in which attacker transmits packets to the vulnerable system. We consider that each packet transmits to the Darknet is an illegal packet. So, the Darknet is effective criteria to explore the scan attack.

Observation Period: At the current time, we use 7/24 Darknet traffic. Our observation to this network area is from 1 November, 2016 to 30 November, 2016.

Our observation on date 30 November, 2016 is described in following Table 6.12.

All details of supporting test model have given in Figure 6.14.

1. With the help of the observation of Botnet and Darknet, an attacker can initiate any DDoS flood attack and generate service unavailability on the site of Cloud Service Provider (CSP).

Table 6.12: The Number of Arrival Packets on Date 30 November, 2016

Time	Number of Arrival Packets
13:45:03 - 13:50:57	69
13:50:58 - 13:55:59	49
13:56:00 - 14:01:23	87
14:01:24 - 14:05:30	127
14:05:31 - 14:10:30	161
14:10:31 - 14:15:30	47
14:15:31 - 14:20:30	120
14:20:31 - 14:25:30	173
14:25:31 - 14:30:30	165
14:30:31 - 14:35:30	67
14:35:31 - 14:40:30	79
14:40:31 - 14:45:30	61

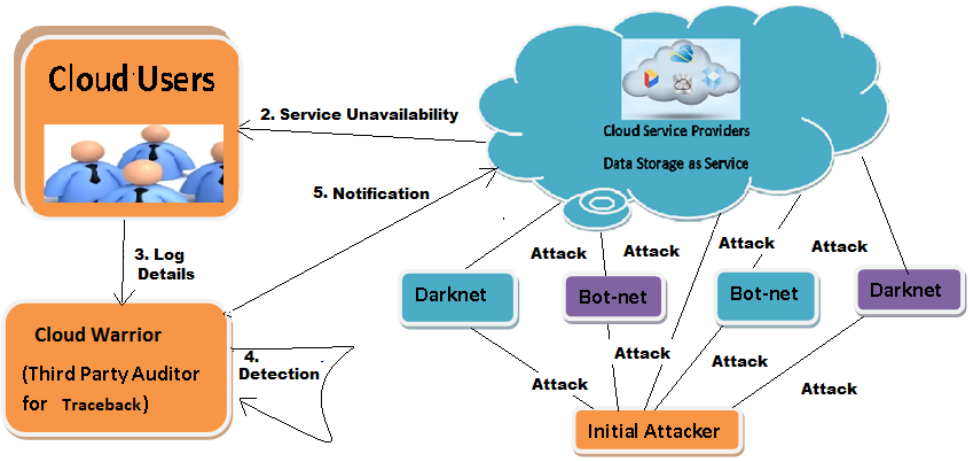


Figure 6.14: Test Model of DDoS Attack Prevention

2. Due to this reason, Cloud users are unable to get services.
3. After service unavailability, Cloud users forward all packets log details to CW, who is a trusted TPA for packet trace-back.
4. It detects the reason and source of attack also in case of DNS and IP spoofing. We efficiently use DNS cache probing for this purpose. It is a well known fact that the most of the network services are used DNS names for identification of their servers. So, DNS resolution is one of the pre-requisite steps to connect server from any client. In DNS cache probing, for each request of DNS name of interest, we examine at regular intervals the cache(s) of the DNS resolver(s) for the network(s) of interest and observe cache hits. For each cache probe, a cooperative resolver reports a hit if it is in the cache or miss otherwise. A cache hit is successful if at least one client made a request for that entry; otherwise it is flushed out. We can efficiently estimate the probes that reveal the sequence of start and end times in the resolver servers. At this time, a direct result sends the combined queries from all clients to the resolver server. With a time to live (TTL) entry for a DNS S, we estimate the aggregate rate γ as follows: we probe the cache resolver at the rate of one probe per TTL. For this, we capture the sequence of start and end times in the resolver cache and for probe p, cache probe time is T_p, resolver return time T_l until the cached entry is expired. Thus, for a TTL, the most recent start time (refresh time) T_r can be calculated with the help of following equation 6.22:

$$T_r = T_p - (TTL - T_l) \tag{6.22}$$

5. Finally, CW notifies exact details of the attack to the CSP.

6.3.4 Port Analysis

We observe the packet traffic pattern on the TCP port numbers 445, 6886, 16888, 11740, 23791, 23830, 23980, and 23981. We also analyze the UDP packet traffic pattern on UDP port numbers 53, 1863, 13347, 20284, 20339, 23981, 31327, 40971, and 42847. We have chosen these specific ports because the most of the services and web traffic are available

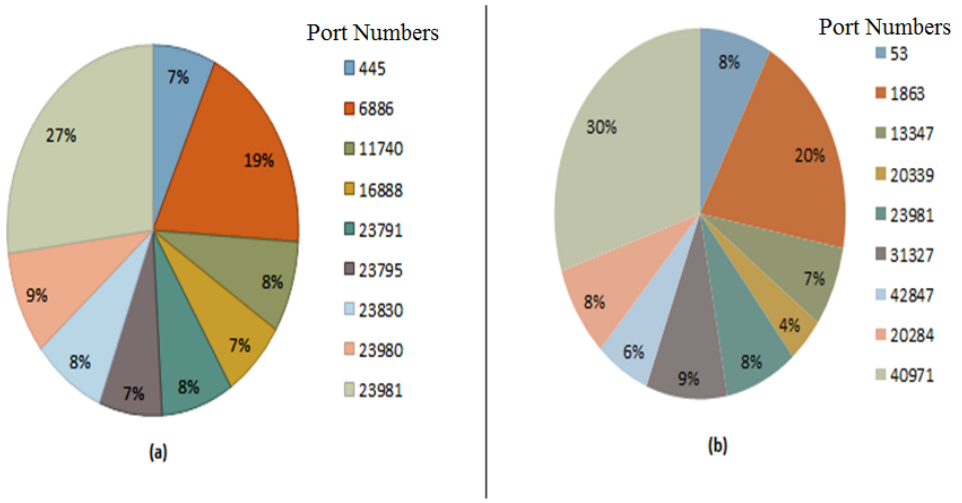


Figure 6.15: Rate of Packets in Different Destination Port (a) TCP (b) UDP

through these ports.

We provide the rate of detected packets in different destination ports in Figure 6.15. The top values of both the graphs represent the vulnerability to attack. Thus, an attacker attempts to exploit this vulnerability and expands the infection. We consider each packet has the certain intention to reach into Darknet area when it has many collaborated s-IPs and the same size of packets.

Currently, in our method, the value of threshold λ set manually for detection of collaborative behaviour. Thus, we manually discover the collaborative behaviour of attack from its packet. The number of detected behaviour of the collaborative packet is 1,078 on TCP and 4,674 on UDP. Thus, the cut rate for used traffic data is 96.56% for TCP and 90.54% for UDP. Using this cut rates we succeeded in reducing many packets.

The maximum execution time for TCP is 117 [milliseconds] and UDP is 25 [milliseconds]. Similarly, average execution time is 109 [milliseconds] and UDP is 16 [milliseconds]. We analyze this traffic pattern in Darknet area. We also need to take care of scale of collaborative behaviour and transmission time of packet on the group of s-IPs and number of d-IPs in the Darknet.

Before the attacks, a threshold has been kept for assessment of the packets. After reaching this threshold, all packets assumed as under attack. A depletion threshold value measures the highest and average value of packets under attack.

To calibrate the Threshold or Tolerance Factor z, we calculate the probability (p) for each

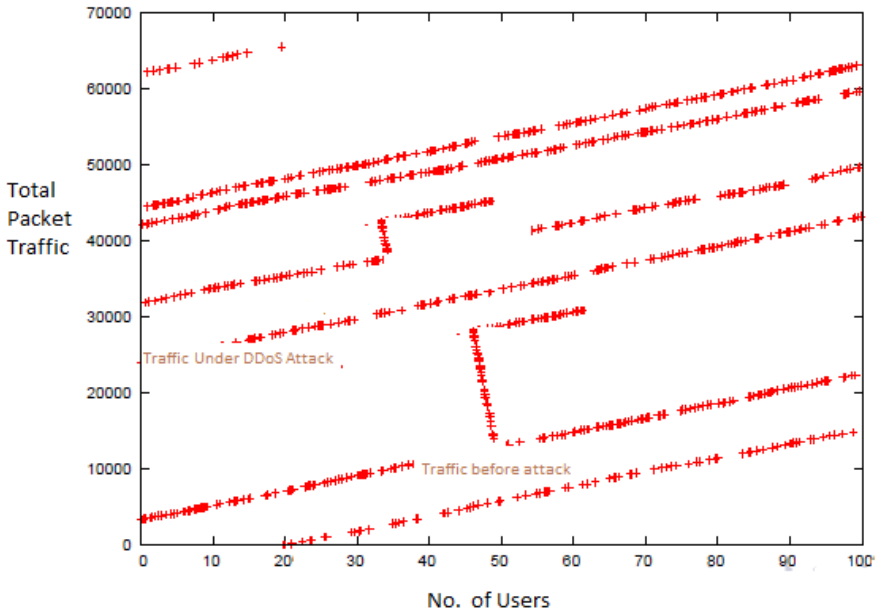


Figure 6.16: Traffic Behavior (a) Before Attack (b) After Attack

source (or destination) address (port) x_i as follows:

$$p(x_i) = \frac{Number\ of\ packets\ with\ x_i\ as\ a\ source\ (or\ destination)\ address\ (Port)}{Total\ Number\ of\ Packets} \tag{6.23}$$

We accumulate this probability for the total number of source (or destination) address (port) as the packet traffic behavior. We observe this packet traffic behavior before and under the traffic. Figure 6.16 shows the traffic behavior under this conditions.

6.3.5 Performance Analysis

The best option to analyze the performance of any method is observing the performance parameters. We have observed different performance parameters of our approach. Table 6.13 describes the parameter assessment of our approach. It demonstrates that when the threshold value λ and the number of total sample increases, then the sensitivity, specificity, and accuracy also increases. Only the false alarm rate have decreased because the number of false positives came down for the total number of samples and the threshold value λ.

Table 6.13: Parameters Assessment of Our Approach

Total Sample	True Positives	True Negatives	False Positives	False Negatives	Sensitivity (%)	Specificity (%)	False Alarm Rate (%)	Accuracy (%)
100	63	26	5	6	91.30	83.87	16.13	89.00
200	123	59	9	9	93.18	86.76	13.24	91.00
300	193	83	11	13	93.69	88.30	11.70	92.00
400	287	93	13	7	97.69	87.74	12.26	95.00
500	383	89	16	12	96.96	84.76	15.24	94.40
600	452	118	19	11	97.62	86.13	13.87	95.00
700	492	183	15	10	98.01	92.42	7.58	96.43
800	557	217	17	9	98.41	92.74	7.27	96.75
900	617	248	21	14	97.78	92.19	7.81	96.11
1000	713	261	10	16	97.81	96.31	3.69	97.40

6.4 Summary

This chapter presents experimental results for our proposed approaches. We have evaluated the proposed authentication scheme with respect to user behaviour and SLA parameters. We achieve on an average of 0.22 FRR, 0.15 URR, and 0.15 RAR for user behaviour parameters and 0.91 TAE, 0.94 RA, 0.96 STR, and 0.99 IP for SLA parameters, respectively. Further, comparative study of existing approaches shows that proposed authentication approach outperforms others with respect to user performances. Our proposed DIV methods are evaluated with different efficiency parameters. Our methods achieve probability of server misbehaviour detection up to 0.95. This is the highest probability of server misbehaviour detection amongst existing methods. We also obtain the highest accuracy and availability in our proposed methods in comparison to existing methods. Moreover, the analysis of our methods by varying different parameters shows that the proposed DIV methods always perform better than the existing methods. Finally, we have prepared a test plan for DDoS attack prevention method. We have analyzed data packet rates at different ports and defined 96.56% and 90.54% of cut rate for TCP and UDP packets. We have also experimented the proposed method with respect to different accuracy parameters. We observed on an average 96.24% of sensitivity, 89.12% specificity, and 94.31% accuracy for our proposed method.

Chapter 7

Conclusion and Future Work

The main objective of our research is to explore efficient methods to address three main security issues in cloud environment. The outcomes of our research are discussed in Chapter 3 to 6 in this thesis. In this chapter, we summarize some salient features of our research contributions. Section 7.1 enlightens over efficient authentication mechanism in cloud environment. In section 7.2, we discuss the important features of the proposed data integrity verification methods. We also manifest over the proposed DDoS attack prevention methods in section 7.3. Finally, we give some future research directions in this area of research in section 7.4.

7.1 Effective Authentication

We propose a novel hierarchical authentication mechanism based on user identities. Multiple users can authenticate with multiple CSPs simultaneously. For this purpose, we maintain a hierarchy of users' identities to generate secret keys for different users. The secret key generation task is delegated to the secret key generator (SKG). The SKG is responsible for the selection of identities at different hierarchical levels and the secure transmission of the secret keys. It also preserves information on the various public parameters corresponding to different CSPs. The secret key generation process goes through a set of hierarchical levels. At each level of hierarchy, an intermediate secret key is generated using a bi-linear pairing between the user identity of that level and the intermediate secret key of the previous level. Further, a set of public parameters are used for effective authentication of multiple cloud

Table 7.1: Assessment of User's Performance

FRR	URR	RAR	UP
0.21	0.19	0.09	91.4
0.19	0.11	0.10	92.9
0.22	0.17	0.08	92
0.23	0.16	0.11	91.2
0.24	0.13	0.15	90.5
0.21	0.12	0.12	91.9
0.25	0.14	0.13	90.8
0.22	0.18	0.11	90.9
0.19	0.15	0.12	91.5
0.23	0.165	0.111	91.37

users with different cloud servers. We measured User Performance (UP) as a weighted sum of Fake Request Rate (FRR), Unauthorized Request Rate (URR), and Resource Affection Rate (RAR) parameters using Eq. 7.1 where w_1, w_2, and w_3 represent the weights corresponding to the respective parameters. The experimental results are reported in Table 7.1. We achieve an average user performance of 91.2% for the proposed authentication system as reported in Table 7.1 .

$$UP = [1 - ((w_1 \times FRR) + (w_2 \times URR) + (w_3 \times RAR))] * 100 \tag{7.1}$$

We also evaluate CSP Performance (CSPP) as a weighted sum of Turn around Efficiency (TE), Resource Availability (RA), Successful Transaction Rate (STR), and Integrity Preservation (IP) using Eq. 7.2 z_1, z_2, z_3, and z_4 represent the weights corresponding to the respective parameters, and the results are reported in Table 7.2. From Table 7.2, we can see that an average CSP performance of $93\ \%$ is achieved in our proposed authentication scheme.

$$CSPP = [(z_1 * TAE) + (z_2 * RA) + (z_3 * STR) + (z_4 * IP)] * 100 \tag{7.2}$$

Moreover, testing under different attacks reveals that the proposed authentication mechanism is robust and resilient under such attacks.

Table 7.2: Assessment of CSP Performance

$\mathbf{TE(R_k)}$	$\mathbf{RA(R_k)}$	$\mathbf{STR(R_k)}$	$\mathbf{IP(R_k)}$	**CSPP**
0.76	0.90	0.917	0.963	91.63
0.84	0.933	0.964	1.000	94.57
0.41	0.914	0.938	0.979	89.68
0.56	0.938	0.967	1.000	93.37
0.54	0.933	0.957	1.000	92.77
0.59	0.975	0.962	1.000	94.26
0.68	0.931	0.986	0.984	94.36
0.71	0.967	0.977	1.000	95.75
0.75	0.978	0.968	0.987	95.58
0.78	0.98	0.969	1.000	96.47

7.2 Data Integrity Verification

In this technique, we propose a public audit structure to provide a privacy preserving audit protocol for verification of cloud data. We explore two approaches for data integrity verification. In the first approach, Combinatorial Batch Codes (CBC), homomorphic tag, and algebraic signature are used for data auditing. This approach operates in two phases. First, users' data are stored into CSPs with the help of a TPA group. At the time of data storing, the CSP generates homomorphic tags which help the TPA to verify users' data without disclosing any information of the stored data in the second phase. In the second phase, the TPA group performs a challenge operation with the CSP storage servers to perform data integrity verification on behalf of cloud users, and the CSP generates proof for this challenge operation. Finally, the proof verification for data auditing is done by the TPA group and the results are notified to the cloud users.

In our second approach, we use the Paillier homomorphic cryptography system for data auditing. Here, a cloud user generates encryption and decryption keys, and encrypts the user data with a multi-power RSA algorithm. This encrypted data is stored into the CSP through the TPA. At the time of verification, CSP performs an assessment process and returns the encrypted responses to the TPA. The TPA decrypts the encrypted responses and verifies the decrypted responses with an assessed file to check whether data integrity is preserved or not. Finally, the TPA forwards the verification results to the cloud user.

Table 7.3: Accuracy Assessment of Our DIV Method 1

Total Sample	**True Positives**	**True Negatives**	**False Positives**	**False Negatives**	**Accuracy (%)**
100	65	28	2	5	93
200	128	63	4	5	95.5
300	198	87	8	7	95
400	287	93	13	7	95
500	389	93	10	8	96.4

The accuracy figures of the proposed approaches with respect to other existing approaches are reported in Table 7.3 and Table 7.4. We achieve an average accuracy of 95% and 92.76% for CBC based and Paillier homomorphic cryptography system based data integrity verification methods, respectively. Further, both the approaches support dynamic data operations with adaptable and useful batch auditing through which various audit sessions for multiple users can be handled simultaneously.

7.3 DDoS Attack Preservation

We propose an effective TPA based trace-back approach for observation of all kinds of packets that reach cloud servers. TPA is used to filter out malicious traffic from legitimate traffic. It also identifies the actual source of DDoS with an accurate prediction model using a trace-back method. It includes three components, namely Internet, front-end server, virtual private cloud. TPA uses the Weibull probability distribution for detecting the source of the TCP flood, the UDP flood, and the ICMP flood attacks. We also provide a service model for fault analysis of each virtual machine to prevent DDoS attacks. We have analyzed the packet

Table 7.4: Parameters Assessment of Our DIV Method 2

Total Sample	**True Positives**	**True Negatives**	**False Positives**	**False Negatives**	**Accuracy %**
100	67	23	5	6	90
200	131	51	9	9	91
300	197	81	10	12	92.67
400	291	92	12	5	95.75
500	383	89	16	12	94.4

traffic pattern on different TCP and UDP ports and evaluated our proposed method with respect to different parameters. The results obtained with our model are shown in Fig. 7.1. It is evident from Fig. 7.1 that the proposed model achieves high sensitivity, specificity, and accuracy with a low false alarm rate.

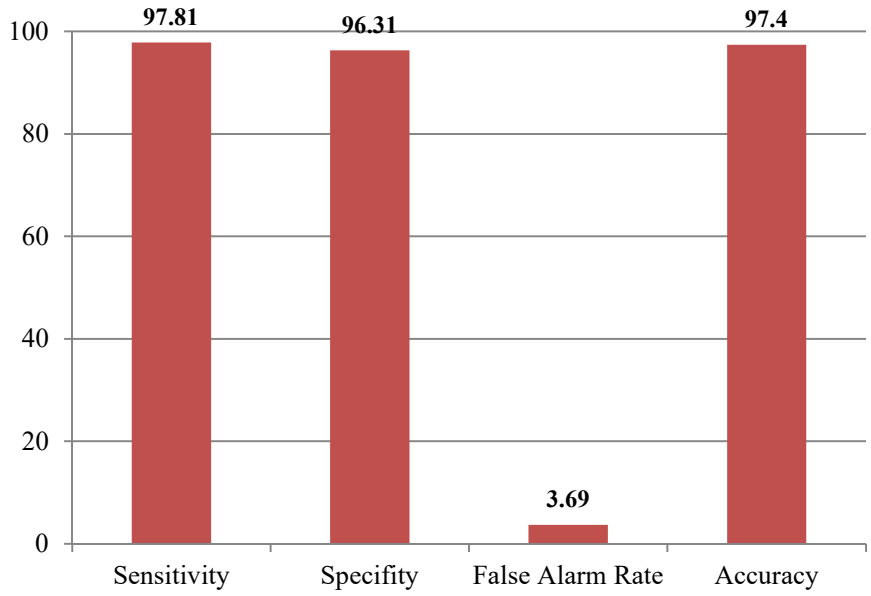


Figure 7.1: Results of the proposed TPA based DDoS Attack Prevention Method with Respect to Different Parameters

7.4 Future Research Direction

Though, we have made significant improvement to solve the security issues in cloud environment, this thesis also delivers various future pathways for research. Some of them are indicated as below.

1. In efficient authentication scheme, one limitation of our approach is that there is a trade-off between the number of hierarchy and time taken to generate secret key. This part can be explored more to generate secret key with less time.
2. We have investigated bi-linear pairing to generate secret key which is dependent on the previous intermediate key. In future, this dependency can be removed using other key generation methods.
3. The proposed authentication scheme does not define any access control mechanism based on user's roles. We can explore this research avenue in near future.

4. In the proposed DIV methods, we have investigated trusted TPA based method. However, end user may not trust on TPA for data integrity verification. Hence, there is a scope to explore DIV methods for un-trusted TPAs.
5. Our proposed DIV methods can successfully handle dynamic data operations. However, one limitation of our methods is that it can not handle these operations simultaneously. We will explore the possibility to perform multiple dynamic data operations simultaneously in future.
6. Both DIV methods are tested on simulation environment with limited size of data. In future, we can test our methods with real cloud environment with large size of data.
7. In our proposed DDoS attack detection method, we use packet trace back method. Sometimes, this method may not detect the source of DDoS attack accurately due to different IP hiding techniques. We will also investigate this issue in our future work.
8. We have collected less data packets to analyse the traffic pattern. We have to explore more to analyse DDoS attack pattern with a large number of data packets in near future.

Bibliography

[1] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," National Institute of Standards and Technology, Tech. Rep., 2011, (Accessed on 20-10-2012).

[2] A. Amies, H. Sluiman, Q. G. Tong, and G. N. Liu, *Developing and Hosting Applications on the Cloud: Develop Hosting Applica Cloud*. IBM Press, 2012.

[3] J. Hurwitz, M. Kaufman, and D. Kirsch, *Hybrid Cloud For Dummies*. John Wiley & Sons, Inc., 2015.

[4] J. A. Olson, "Data as a Service: Are We in the Clouds?" *Journal of Map & Geography Libraries*, vol. 6, no. 1, pp. 76–78, 2009.

[5] Microsoft, "What is SaaS? Software as a Service," 2012, (Accessed on 20-10-2012). [Online]. Available: https://azure.microsoft.com/en-in/overview/what-is-saas/

[6] S. Zhang, S. Zhang, X. Chen, and X. Huo, "Cloud Computing Research and Development Trend," in *Proceedings of the Second International Conference on Future Networks (ICFN'10)*, Sanya, China, 2010, pp. 93–97.

[7] C. Gong, J. Liu, Q. Zhang, H. Chen, and Z. Gong, "The Characteristics of Cloud Computing," in *Proceedings of the* 39^{th} *International Conference on Parallel Processing Workshops (ICPPW '10)*, San Diego, CA, USA, 2010, pp. 275–279.

[8] D. He, N. Kumar, M. K. Khan, L. Wang, and J. Shen, "Efficient Privacy-Aware Authentication Scheme for Mobile Cloud Computing Services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1621–1631, 2018.

[9] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in *Proceedings of the* 14^{th} *ACM Conference on Computer and Communications Security*, Alexandria, USA, 2007, pp. 598–609.

[10] S. Ranjan, R. Swaminathan, M. Uysal, A. Nucci, and E. Knightly, "DDoS-Shield: DDoS-Resilient Scheduling to Counter Application Layer Attacks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 1, pp. 26–39, 2009.

[11] S. Chandrasekhar and M. Singhal, "Efficient and Scalable Query Authentication for Cloud-Based Storage Systems with Multiple Data Sources," *IEEE Transactions on Services Computing*, vol. 10, no. 4, pp. 520–533, 2017.

[12] V. Odelu, A. K. Das, S. Kumari, X. Huang, and M. Wazid, "Provably Secure Authenticated Key Agreement Scheme for Distributed Mobile Cloud Computing Services," *Future Generation Computer Systems*, vol. 68, pp. 74–88, 2017.

[13] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and Efficient Provable Data Possession," in *Proceedings of the* 4^{th} *International Conference on Security and Privacy in Communication Netowrks*, 2008, pp. 9:1–9:10.

[14] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," in *Proceedings of the* 16^{th} *ACM Conference on Computer and Communications Security*, Chicago, Illinois, USA, 2009, pp. 213–222.

[15] A. Juels and B. S. Kaliski, "PORs: Proofs of Retrievability for Large Files," in *Proceedings of the* 14^{th} *ACM Conference on Computer and Communications Security*, Alexandria, USA, 2007, pp. 584–597.

[16] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, Melbourne, Australia, 2008, pp. 90–107.

[17] M. Etemad and A. Küpçü, "Transparent, Distributed, and Replicated Dynamic Provable Data Possession," in *Proceedings of the International Conference on Applied Cryptography and Network Security*, Bogota, USA, 2013, pp. 1–18.

[18] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy Preserving Public Auditing for Data Storage Security in Cloud Computing," in *Proceedings of the IEEE Infocom*, San Diego, USA, 2010, pp. 1–9.

[19] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[20] Z. Hao, S. Zhong, and N. Yu, "A Privacy-preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 9, pp. 1432–1437, 2011.

[21] R. Braga, E. Mota, and A. Passito, "Lightweight DDoS Flooding Attack Detection Using NOX/OpenFlow," in *Proceedings of the* 35^{th} *IEEE Conference on Local Computer Networks (LCN)*, OsnabrÃijck, Germany, 2010, pp. 408–415.

[22] Y. Choi, "Implementation of Content-Oriented Networking Architecture (CONA): A Focus on DDoS Countermeasure," in *Proceedings of the European NetFPGA developers workshop*, Wuhan, China, 2010, pp. 35–38.

[23] R. Lua and K. C. Yow, "Mitigating DDoS Attacks with Transparent and Intelligent Fast-Flux Swarm Network," *IEEE Network*, vol. 25, no. 4, pp. 28 – 33, 2011.

[24] J. Mirkovic and P. Reiher, "A Taxonomy of DDoS Attack and DDoS Defense Mechanisms," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 39–53, 2004.

[25] W. Litwin and T. Schwarz, "Algebraic Signatures for Scalable Distributed Data Structures," in *Proceedings of the* 20^{th} *International Conference on Data Engineering*, Boston, USA, 2004, pp. 412–423.

[26] L. Chen, "Using Algebraic Signatures to Check Data Possession in Cloud Storage," *Future Generation Computing Systems*, vol. 29, no. 7, pp. 1709–1715, 2013.

[27] Q. Do, B. Martini, and K. K. R. Choo, "A Cloud-Focused Mobile Forensics Methodology," *IEEE Cloud Computing*, vol. 2, no. 4, pp. 60–65, 2015.

[28] Dropbox, "Put Your Creative Energy to Work, with Dropbox," 2017, (Accessed on 20-10-2017). [Online]. Available: https://www.dropbox.com/

[29] I. S. Sette, D. W. Chadwick, and C. A. G. Ferraz, "Authorization Policy Federation in Heterogeneous Multicloud Environments," *IEEE Cloud Computing*, vol. 4, no. 4, pp. 38–47, 2017.

[30] Q. Jiang, J. Ni, J. Ma, L. Yang, and X. Shen, "Integrated Authentication and Key Agreement Framework for Vehicular Cloud Computing," *IEEE Network*, vol. 32, no. 3, pp. 28–35, 2018.

[31] Q. Jiang, J. Ma, and F. Wei, "On the Security of a Privacy-Aware Authentication Scheme for Distributed Mobile Cloud Computing Services," *IEEE Systems Journal*, vol. 12, no. 2, pp. 2039–2042, 2018.

[32] R. Amin, N. Kumar, G. P. Biswas, R. Iqbal, and V. Chang, "A Light Weight Authentication Protocol for IoT-enabled Devices in Distributed Cloud Computing Environment," *Future Generation Computer Systems*, vol. 78, pp. 1005–1019, 2018.

[33] K. Benzekki, A. E. Fergougui, and A. E. ElAlaoui, "A Context-Aware Authentication System for Mobile Cloud Computing," *Procedia Computer Science*, vol. 127, pp. 379–87, 2018.

[34] F. Wu, X. Li, L. Xu, S. Kumari, and A. K. Sangaiah, "A Novel Mutual Authentication Scheme with Formal Proof for Smart Healthcare Systems Under Global Mobility Networks Notion," *Computers & Electrical Engineering*, vol. 68, pp. 107–11, 2018.

[35] I. Butun, M. E. Kantarci, B. Kantarci, and H. Song, "Cloud-Centric Multi-Level Authentication as a Service for Secure Public Safety Device Networks," *IEEE Communications Magazine*, vol. 54, no. 4, pp. 47–53, 2016.

[36] R. M. Daniel, E. B. Rajsingh, and S. Silas, "A Forward Secure Signcryption Scheme with Ciphertext Authentication for E-payment Systems Using Conic Curve Cryptography," *Journal of King Saud University - Computer and Information Sciences*, 2018, available online. [Online]. Available: https://doi.org/10.1016/j.jksuci.2018.02.004

[37] C. Zhu, H. Nicanfar, V. C. M. Leung, and L. T. Yang, "An Authenticated Trust and Reputation Calculation and Management System for Cloud and Sensor Networks Integration," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 1, pp. 118–131, 2015.

[38] S. Grzonkowski and P. M. Corcoran, "Sharing Cloud Services: User Authentication for Social Enhancement of Home Networking," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1424–1432, 2011.

[39] H. Liu, H. Ning, Q. Xiong, and L. T. Yang, "Shared Authority Based Privacy-Preserving Authentication Protocol in Cloud Computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 1, pp. 241–251, 2015.

[40] Z. Liu, H. Yan, and Z. Li, "Server-aided Anonymous Attribute-based Authentication in Cloud Computing," *Future Generation Computer Systems*, vol. 52, pp. 61–66, 2015, special Section: Cloud Computing: Security, Privacy and Practice.

[41] A. Kumar and H. Om, "An Improved and Secure Multiserver Authentication Scheme based on Biometrics and Smartcard," *Digital Communications and Network*, vol. 4, no. 1, pp. 27–38, 2018.

[42] C. Guo, N. Luo, M. Z. A. Bhuiyan, Y. Jie, Y. Chen, B. Feng, and M. Ala, "Key-Aggregate Authentication Cryptosystem for Data Sharing in Dynamic Cloud Storage," *Future Generation Computer Systems*, vol. 84, pp. 190–199, 2018.

[43] R. Saxena and S. Dey, "Collaborative Approach for Data Integrity Verification in Cloud Computing," in *Proceedings of the Second International Conference on Recent Trends in Computer Networks and Distributed Systems Security , SNDS*, Trivandrum, India, 2014, pp. 1–15.

[44] F. Sebé, J. Domingo-Ferre, A. Martinez-Balleste, Y. Deswarte, and J. J. Quisquater, "Efficient Remote Data Possession Checking in Critical Information Infrastructures," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 8, pp. 1034–1038, 2008.

[45] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," *ACM Transactions on Information and System Security (TISSEC)*, vol. 17, no. 4, pp. 15:1–15:29, 2015.

[46] L. Chen, "Using Algebraic Signatures to Check Data Possession in Cloud Storage," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1709–1715, 2013.

[47] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of Retrievability via Hardness Amplification," in *Proceedings of the Theory of Cryptography Conference*, Baltimore, USA, 2009, pp. 109–127.

[48] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of Retrievability: Theory and Implementation," in *Proceedings of the ACM workshop on Cloud computing Security*, Chicago, USA, 2009, pp. 43–54.

[49] K. D. Bowers, A. Juels, and A. Oprea, "HAIL: A High-Availability and Integrity Layer for Cloud Storage," in *Proceedings of the* 16^{th} *ACM conference on Computer and Communications Security*, Toronto, Canada, 2009, pp. 187–198.

[50] Y. Zhu, H. Hu, G. J. Ahn, and M. Yu, "Cooperative Provable Data Possession for Integrity Verification in Multicloud Storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 12, pp. 2231–2244, 2012.

[51] S. Rizvi, A. Razaque, and K. Cover, "Third-Party Auditor (TPA): A Potential Solution for Securing a Cloud Environment," in *Proceedings of the* 2^{nd} *International Conference on Cyber Security and Cloud Computing*, Cambridge, United Kingdom, 2015, pp. 31–36.

[52] S. Z. Mei, C. Liu, Y. Cheng, J. Wu, and Z. Wang, "TETPA: A Case for Trusted Third Party Auditor in Cloud Environment," in *Proceedings of the IEEE Conference Anthology*, Wuhan, China, 2013, pp. 1–4.

[53] K. Huang, M. Xian, S. Fu, and J. Liu, "Securing the Cloud Storage Audit Service: Defending Against Frame and Collude Attacks of Third Party Auditor," *IET Communications*, vol. 8, no. 12, pp. 2106–2113, 2014.

[54] Y. Zhang, C. Xu, S. Yu, H. Li, and X. Zhang, "SCLPV: Secure Certificateless Public Verification for Cloud-based Cyber-Physical-Social Systems Against Malicious Auditors," *IEEE Transactions on Computational Social Systems*, vol. 2, no. 4, pp. 159–170, 2015.

[55] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-Preserving Public Auditing Protocol for Low-Performance End Devices in Cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, 2016.

[56] X. Zhang, C. Xu, Y. Zhang, X. Zhang, and J. Wen, "Insecurity of a Public Proof of Cloud Storage from Lattice Assumption," *Chinese Journal of Electronics*, vol. 26, no. 1, pp. 88–92, 2017.

[57] T. Xiang, X. Li, F. Chen, Y. Yang, and S. Zhang, "Achieving Verifiable, Dynamic and Efficient Auditing for Outsourced Database in Cloud," *Journal of Parallel and Distributed Computing*, vol. 112, no. 1, pp. 97–107, 2018.

[58] J. Yu, K. Ren, and C. Wang, "Enabling Cloud Storage Auditing with Verifiable Outsourcing of Key Updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362–1375, 2016.

[59] S. B. Lee, M. S. Kang, and V. D. Gligor, "CoDef: Collaborative Defense Against Large-Scale Link-Flooding Attacks," in *Proceedings of the* 9^{th} *ACM Conference on Emerging Networking Experiments and Technologies*, Berkeley, USA, 2013, pp. 417–428.

[60] B. Sieklik, R. Macfarlane, and W. J. Buchanan, "Evaluation of TFTP DDoS Amplification Attack," *Computers & Security*, vol. 57, pp. 67–92, 2016.

[61] P. J. Criscuolo, "Distributed Denial of Service: Trin00, Tribe Flood Network, Tribe Flood Network 2000, and Stacheldraht Ciac-2319," California University Livemore Radiation Lab, Tech. Rep., 2000, (Accessed on 10-01-2018).

[62] D. Dittrich, "The Tribe Flood Network, Distributed Denial of Service Attack Tool," 1991, (Accessed on 20-10-2018). [Online]. Available: http://staff.washington.edu/dittrich/misc/tfn.analysis

[63] D. Dittrich, "The 'Stacheldraht' Distributed Denial of Service Attack Tooll," 1991, (Accessed on 20-10-2018). [Online]. Available: http://staff.washington.edu/dittrich/misc/stacheldraht.analysis

[64] G. W. D. Dittrich, S. Dietrich, and N. Long, "The 'Mstream' Distributed Denial of Service Attack Tool," 2000, (Accessed on 20-10-2018). [Online]. Available: http://staff.washington.edu/dittrich/misc/mstream.analysis.txt.

[65] S. Dietrich, N. Long, and D. Dittrich, "Analyzing Distributed Denial of Service Tools: The Shaft Case," in *Proceedings of the 14th USENIX Conference on System Administration*, New Orleans, Louisiana, 2000, pp. 329–340.

[66] B. Hancock, "Trinity v3, a DDoS Tool, Hits the Streets," *Computers & Security*, vol. 19, no. 7, pp. 574–574, 2000.

[67] Bysin, "Sourcecode: Knight.c," 2001, (Accessed on 20-10-2018). [Online]. Available: http://packetstormsecurity.org/distributed/knight.c

[68] J. Nazario, "Blackenergy DDos Bot Analysis," 2007, (Accessed on 20-10-2018). [Online]. Available: http://atlas-public.ec2.arbor.net/docs/BlackEnergy+DDoS+Bot+Analysis.pdf

[69] A. Sert, "DDoS and Security Reports: The Arbor Networks Security Blog," 2011, (Accessed on 15-10-2018). [Online]. Available: http://ddos.arbornetworks.com/2012/02/ddos-tools/

[70] A. Shameli-Sendi, M. Pourzandi, M. Fekih-Ahmed, and M. Cheriet, "Taxonomy of Distributed Denial of Service Mitigation Approaches for Cloud Computing," *Journal of Network and Computer Applications*, vol. 58, pp. 165–179, 2015.

[71] C. YuHunag, T. MinChi, C. YaoTing, C. YuChieh, and C. YanRen, "A Novel Design for Future On-demand Service and Security," in *Proceedings of the* 12^{th} *IEEE International Conference on Communication Technology (ICCT)*, Nanjing City, China, 2010, pp. 385–388.

[72] G. Yao, J. Bi, and P. Xiao, "Source Address Validation Solution with OpenFlow/NOX architecture," in *Proceedings of the* 19^{th} *IEEE International Conference on Network Protocols (ICNP)*, Vancouver, Canada, 2011, pp. 7–12.

[73] W. Dou, Q. Chen, and J. Chen, "A Confidence-based Filtering Method for DDoS Attack Defense in Cloud Environment," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1838–1850, 2013.

[74] S. Shin, P. A. Porras, V. Yegneswaran, M. W. Fong, G. Gu, and M. Tyson, "FRESCO: Modular Composable Security Services for Software-Defined Networks," in *Proceedings of the* 20^{th} *Annual Network & Distributed System Security Symposium*, San Diego, USA, 2013, pp. 7–15.

[75] S. T. Zargar and J. B. Joshi, "A Collaborative Approach to Facilitate Intrusion Detection and Response Against DDoS Attacks," in *Proceedings of the* 6^{th} *International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*, Washington, USA, 2010, pp. 1–8.

[76] S. Yu, Y. Tian, S. Guo, and D. O. Wu, "Can We Beat DDoS Attacks in Clouds?" *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2245–2254, 2014.

[77] R. Saxena and S. Dey, "Cloud Shield: Effective Solution for DDoS in Cloud," in *Proceedings of the* 8^{th} *International Conference on Internet and Distributed Computing Systems IDCS*, 2015, pp. 3–10.

[78] K. Kalkan and F. Alagöz, "A Distributed Filtering Mechanism Against DDoS Attacks: ScoreForCore," *Computer Networks*, vol. 108, pp. 199–209, 2016.

[79] Z. Wang, "An Elastic and Resiliency Defense Against DDoS Attacks on the Critical DNS Authoritative Infrastructure," *Journal of Computer and System Sciences*, vol. 99, pp. 1–26, 2019.

[80] K. Singh, P. Singh, and K. Kumar, "A systematic review of IP traceback schemes for denial of service attacks," *Computers & Security*, vol. 56, pp. 111–139, 2016.

[81] K. Singh, P. Singh, and K. Kumar, "Application Layer HTTP-GET Flood DDoS Attacks: Research Landscape and Challenges," *Computers & Security*, vol. 65, pp. 344–372, 2017.

[82] I. Nunes, F. Schardong, and A. S. Filho, "BDI2DoS: An Application Using Collaborating BDI Agents to Combat DDoS Attacks," *Journal of Network and Computer Applications*, vol. 84, pp. 14–24, 2017.

[83] G. Somani, M. S. Gaur, D. Sanghi, and M. Conti, "DDoS Attacks in Cloud Computing: Collateral Damage to Non-targets," *Computer Networks*, vol. 109, pp. 157–171, 2016.

[84] B. Wang, Y. Zheng, W. Lou, and Y. T. Hou, "DDoS Attack Protection in the Era of Cloud Computing and Software-Defined Networking," *Computer Networks*, vol. 81, pp. 308–319, 2015.

[85] B. Lynn, "On the Implementation of Pairing-based Cryptosystems," Ph.D. dissertation, Stanford University Stanford, California, 2007.

[86] T. S. Schwarz and E. L. Miller, "Store, Forget, and Check: Using Algebraic Signatures to Check Remotely Administered Storage," in *Proceedings of the* 26^{th} *IEEE International Conference on Distributed Computing Systems, (ICDCS 2006)*, Lisboa, Portugal, 2006, pp. 12–12.

[87] D. R. Stinson, R. Wei, and M. B. Paterson, "Combinatorial Batch Codes," *Advances in Mathematics of Communications*, vol. 3, no. 1, pp. 13–27, 2009.

[88] C. Bujtás and Z. Tuza, "Optimal Combinatorial Batch Codes Derived from Dual Systems," *Miskolc Mathematical Notes*, vol. 12, no. 1, pp. 11–23, 2011.

[89] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Batch Codes and Their Applications," in *Proceedings of the* 36^{th} *Annual ACM Symposium on Theory of Computing*, Chicago, USA, 2004, pp. 262–271.

[90] P. Paillier, "Public-key Cryptosystems-based on Composite Degree Residuosity Classes," in *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, Prague, Czech Republic, 1999, pp. 223–238.

[91] D. Boneh, G. Durfee, and N. H. Graham, "Factoring N = $p^r q$ for Large r," in *Proceedings of the* 19^{th} *Annual International Cryptology Conference*, Santa Barbara, USA, 1999, pp. 326–337.

[92] X. Wang, G. Xu, M. Wang, and X. Meng, *Mathematical Foundations of Public Key Cryptography*. CRC Press, 2015.

[93] R. C. Merkle, "Protocols for Public Key Cryptosystems," in *Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, USA, 1980, pp. 122–122.

[94] R. Saxena and S. Dey, "Cloud Audit: A Data Integrity Verification Approach for Cloud Computing," *Procedia Computer Science*, vol. 89, pp. 142–151, 2016.

[95] Cloudera, "Cloudera Downloads Get Started With Hadoop @ONLINE," 2014, (Accessed on 17-02-2015). [Online]. Available: http://www.cloudera.com/content/cloudera/en/downloads.html

[96] XenServer, "Download XenServer 6.2 @ONLINE," 2014, (Accessed on 17-02-2015). [Online]. Available: http://xenserver.org/open-source-virtualization-download.html

[97] C. Siaterlis, V. Maglaris, and P. Roris, "A Novel Approach for a Distributed Denial of Service Detection Engine," in *HP Open View University Association Conference*, Athens, Greece, 2003.

[98] M. A. Guth, "A Probabilistic Foundation for Vagueness and Imprecision in Fault-Tree Analysis," *IEEE Transactions on Reliability*, vol. 40, no. 5, pp. 563–571, 1991.

[99] C. S. V. Maglaris, "One Step Ahead to Multisensor Data Fusion for DDoS Detection," *Journal of Computer Security*, vol. 13, no. 5, pp. 779–806, 2005.

[100] C. D. Lai, M. Xie, and D. N. P. Murthy, "A Modified Weibull Distribution," *IEEE Transactions on Reliability*, vol. 52, no. 1, pp. 33–37, 2003.

[101] M. S. Specht and R. B. Lee, "Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures," in *Proceedings of the ISCA* 17^{th} *International Conference on Parallel and Distributed Computing Systems*, San Francisco, USA, 2004, pp. 543–550.