

**DESIGN OF HEURISTIC HOMOMORPHIC  
TECHNIQUE FOR DISTRIBUTED MULTI-TENANT  
CLOUD ENVIRONMENT**

Thesis Submitted for the Award of the Degree of

**DOCTOR OF PHILOSOPHY**  
in  
**Computer Applications**

By  
**Pooja Dhiman**

**(41801027)**

**Supervised by**

**Dr. Santosh Kumar Henge**

**Co-Supervised by**

**Dr. Manmohan Sharma**




**LOVELY PROFESSIONAL UNIVERSITY**  
**PUNJAB**  
**2022**

### **Candidate's Declaration**

I now declare that the thesis entitled "**Design of Heuristic Homomorphic Technique for Distributed Multi-Tenant Cloud Environment**" is submitted to the Department of Research, Lovely Professional University, Phagwara, Punjab, to fulfill the requirements for the award of Degree of Doctor of Philosophy in Computer Applications. It is a bonafide record of my own and original research work carried under the supervision of Dr. Santosh Kumar Henge and Dr. Manmohan Sharma, School of Computer Applications, Lovely Professional University, Phagwara.

I further declare that the matter presented in this thesis has not been submitted for the award of any other degree/diploma of this University or any other University/Institute.

  
**(Pooja Dhiman)**  
**Research Scholar**

## Certificate

I certify that the candidate carried out the work incorporated in the thesis “Design of Heuristic Homomorphic Technique for Distributed Multi-Tenant Cloud Environment” submitted by Pooja Dhiman under my supervision/guidance. To the best of my knowledge:

- i. The candidate has not submitted the same research work to any other institution for any degree/diploma, Associateship, Fellowship, or other similar titles.
- ii. The thesis submitted is a record of original research work done by the Research Scholar during the period of study under my supervision, and
- iii. The thesis represents independent research work on the part of the Research Scholar.

Supervised by



**(Dr. Santosh Kumar Henge)**  
Associate Professor  
Lovely Professional University  
Phagwara  
Punjab

Co-Supervised by



**(Dr. Manmohan Sharma)**  
Professor  
Lovely Professional University  
Phagwara  
Punjab

## Letter of Candidacy



**L**OVELY  
**P**ROFESSIONAL  
**U**NIVERSITY

Centre for  
Research Degree Programmes

*LPU/CRDP/PHD/EC/20200504/000950*  
Dated: 19 May 2021

Pooja Dhiman  
Registration Number: 41801027  
Programme Name: Doctor of Philosophy (Computer Applications)

**Subject: Letter of Candidacy for Ph.D.**

Dear Candidate,

We are very pleased to inform you that the Department Doctoral Board has approved your candidacy for the Ph.D. Programme on 27 Feb 2020 by accepting your research proposal entitled: "Design of heuristics Homomorphic technique for distributed Multi-tenant cloud environment" under the supervision of Dr. Santosh Kumar Henge.

As a Ph.D. candidate you are required to abide by the conditions, rules and regulations laid down for Ph.D. Programme of the University, and amendments, if any, made from time to time.

We wish you the very best!!

In case you have any query related to your programme, please contact Centre of Research Degree Programmes.

Head  
Centre for Research Degree Programmes

Note:-This is a computer generated certificate and no signature is required. Please use the reference number generated on this certificate for future conversations.

---

Jalandhar-Delhi G.T.Road, Phagwara, Punjab (India) - 144411  
Ph : +91-1824-444594 E-mail : dr@lpu.co.in website : www.lpu.in

## **Acknowledgment**

I express my sincere thanks and deep gratitude to my supervisor, Dr. Santosh Kumar Henge for his valuable support and efforts. He has truly supported and guided me in the right direction which helped in the fulfilment of the research work. He has always motivated me in writing quality research papers. I express my gratitude to my co-supervisor Dr. Manmohan Sharma for unwavering support, suggestions, and guidance. I express my gratitude to them for their valuable contribution to the progress and development of my research. They possess an immense breadth and depth of knowledge coupled with great talent and enthusiasm for research. They have taught me how to undertake research work and communicate effectively in the form of a research paper. Their active participation, untiring efforts, affection, guidance, and approach have brought this work to the present stage.

I received a lot of support from many researchers, academicians, and professors of other universities as well. I am incredibly helpful for their effort in analysing the proposed work and providing me with an environment for implementing the work.

I extend my sincere thanks to my family and friends, and especially my father Mr. J.P. Dhiman for his moral support and motivation. He taught me how to patiently handle unfavourable situations and maintain balance in life. I deeply thank my husband Mr. Vikram Arora for supporting me in completing my research work.



**Pooja Dhiman**

## ABSTRACT

Cloud computing has completely altered the delivery of IT services to the clients. It eliminates the requirement of setting up of high-cost computing infrastructure. The ability to scale up and down the services according to the requirement of the client makes it flexible and easy to use. Cloud computing provides various hosting and storage services on the internet. Cloud service providers (CSPs) provide privacy by segregating clients' data and resources into tenants. A tenant is an isolated container specifically designed for each client; it can be an organization, a department of an organization, or an individual. Multi-tenancy is the key element in both public and private spaces; privacy protection is the main concern in collaboration among these tenants.

Authentication, authorization, and data access control are essential to maintain the security and privacy of the data. Most of secure systems have integrated a dual-mode authentication process with various control logs in an enterprise-level multi-tenancy environment, but the dual authentication process may not secure the data from upcoming threats and researchers should consider additional layers in the authentication process.

The Fully Homomorphic encryption algorithm encrypts the data on the cloud in a secure way. This scheme ensures data computation on the encrypted state without the need of decrypting the data.

A hybrid approach using EHC (Enhanced Homomorphic Cryptosystem) and BGV (Brakerski-Gentry-Vaikuntanathan) is designed based on the user's role for the proposed model. EHC requires less power and the storage requirement is comparatively less. It is indistinguishable under chosen-ciphertext attack, but it supports implicating the static data. BGV manages ciphertext by reducing noise, compatible with both static and dynamic data, but it's weak in proving the security of CCA.

EHC and BGV can be used in combination to cover up the disadvantages of both algorithms resulting in forming a better algorithm with more advantages and more security. This research considers dependable and non-dependable parameters and

factors to secure the tenants using the EHC-BGV FHE hybrid methodology, resulting in a multi-factor authentication-authorization process. The total number of keys used by BGV is two, one private and one public key while EHC generates one public key and three private keys.

A token is generated for the security of internal users that will expire after some period of time and based on the category of the outsider users; keys are generated. EHC encryption scheme generates the token and keys for most of the insider and outsider user categories while BGV is used to generate a hybrid key for one specific user category that is for partial users only. The highest security is set for the trusted category of outsider tenants in an outsider environment. An additional key OOTP (On-demand one time password) is generated in case of an attack; upon which self-key mutation takes place. This key is added apart from password, salting, and OTP authentication parameters. It adds an additional layer of protection to the cloud data.

For other tenant categories, that is, for an un-trusted user or tenant, two user categories are used; Partial and Guest or Anonymous user. For partial users, the hybrid key is generated which is based on some production rules and the session will expire after some number of days. The guest user is considered as most risky tenant type since a guest or anonymous user does not provide much details before accessing a file. Hence, they are given least access to the resources and the session will expire after a number of hours. In this way, security is applied for different layers of tenants and customized access is given to the tenants according to their roles.

In our proposed model, the Fully Homomorphic algorithm schemes and parity mapping with key mutation technique are used. Key mutation is implemented only in case of an attack. When an attacker tries to intercept the transmission channel, self-key mutation takes place and shuffling of bits is done which is based on the production rules or parity rules.

The proposed approach has tested on various distributed cloud servers with 223 end-users by the integration of seventeen multitenant, twelve head-tenants, and seven enterprise levels and achieved a 96 percent of success rate. Self-key mutation is

implemented in case of an attacking scenario. The attack is identified by checking the user's authorization from the dictionary, which is designed at the time of registration of the user or tenant. The dictionary contains the user's information like IP address, name and location. If the attacker tries to access the data, IP is matched from the dictionary, if the match is not found, self-key mutation occurs. A rule is selected from 56 rules which are already designed randomly. The random selection of mutation rule makes it complex for the attackers to get access to the data.

The proposed blended model is efficient to prevent the data from ciphertext attacks and achieved more success rate for transmitting the data between various multi-tenants based on the user-role-user-type of enterprise cloud servers.



## Table of Contents

<b>Candidate's Declaration</b> .....	<b>ii</b>
<b>Certificate</b> .....	<b>iii</b>
<b>Letter of Candidacy</b> .....	<b>iv</b>
<b>Acknowledgment</b> .....	<b>v</b>
<b>ABSTRACT</b> .....	<b>vi</b>
<b>List of Tables</b> .....	<b>xii</b>
<b>List of Figures</b> .....	<b>xv</b>
<b>Annexure I</b> .....	<b>xvii</b>
List of Abbreviation .....	xvii
<b>Chapter-1</b> .....	<b>1</b>
<b>Introduction and Overview of Cloud Computing</b> .....	<b>1</b>
1.1 Overview of Cloud Computing .....	1
1.1.1 Cloud Service Delivery Models .....	2
1.1.2 Cloud Deployment Models.....	2
1.2 Multi-Tenancy .....	3
1.2.1 SaaS based Multi-Tenancy Challenges and Concerns .....	3
1.3 Virtualization .....	4
1.4 Basic types of attacks.....	5
1.5 Homomorphic Design .....	5
1.5.1 Functions of Homomorphic Encryption .....	7
1.5.2 Homomorphic Encryption Schemes.....	8
1.5.3 Properties of Homomorphic Encryption .....	9
1.6 Homomorphic techniques for securing the data with data access control .....	10
1.7 Analysis of FHE Schemes (BGV, EHC, AHEE, NEHE) .....	11
1.7.1 BGV (Brakerski-Gentry-Vaikuntanathan) .....	11
1.7.2 EHC (Enhanced Homomorphic Cryptosystem) .....	12
1.7.3 AHEE (Algebra Homomorphic Encryption scheme based on updated Elgamal).....	13
1.7.4 NEHE (Non-interactive exponential Homomorphic encryption algorithm) ....	14
1.7.5 Detailed Analysis of various FHE Schemes (NEHE, BGV, AHEE, EHC).....	16
1.7.6 Selection criteria of BGV and EHC FHE schemes .....	17

1.8	EHC, BGV and Hybrid EHC-BGV homomorphic algorithm .....	18
1.9	Brief summary of this chapter .....	19
<b>Chapter-2 .....</b>		<b>20</b>
<b>Literature Review .....</b>		<b>20</b>
2.1	Analysis of literature.....	20
2.2	Comparison of past proposed and proposed model complexities based on number of computations.....	38
2.3	Issues with current methodologies and Research gaps.....	39
2.4	Addressing the issues.....	40
2.5	Brief summary of this chapter .....	41
<b>Chapter-3 .....</b>		<b>42</b>
<b>Methodology of the proposed research work .....</b>		<b>42</b>
3.1	Objectives of the proposed work .....	42
3.2	Implementation of Ciphertext Policy Attribute-Based Encryption (CP-ABE) Algorithm .....	42
3.3	Designing the multi-tenant multiple-enterprise model.....	46
3.4	FHE Blended Schemes EHC and BGV based Environment: .....	48
3.5	Automated Key Filter and Bit-Mapping Techniques using Hybrid EHC-BGV Homomorphic Algorithm.....	49
3.6	Secure Token and Key Implications based on Dependable and Non-Dependable Factors....	52
3.7	Algorithm for generating the token/keys based on the user-role-type .....	55
3.8	Experimental Scenario and Analysis: .....	57
3.8.1	Attacking scenario.....	57
3.8.2	Self-key mutation.....	58
3.8.3	Working of self-key mutation .....	59
3.9	Brief summary of this chapter .....	79
<b>Chapter-4 .....</b>		<b>80</b>
<b>Experimental Analysis and Results .....</b>		<b>80</b>
4.1	Experimental setup.....	80
4.2	Creating a multitenant environment on CloudSim and generating the keys using EHC algorithm .....	84
4.2.1	Token generation.....	85
4.2.2	Snip for token generation on email id.....	85
4.2.3	Storage requirement by EHC .....	86

4.3	Test case analysis.....	87
4.3.1	Implications of EHC algorithm based key generation time, encryption time and decryption time.....	87
4.3.2	Implications of Hybrid EHC-BGV Homomorphic based key generation time, encryption time and decryption time .....	89
4.3.3	Implications of Hybrid EHC-BGV Homomorphic based automated key filter scenario with bits-shuffling mechanism.....	91
4.3.4	Key generation time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size.....	92
4.3.5	Encryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size.....	92
4.3.6	Decryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size.....	93
4.3.7	Description of Comparative analysis of proposed hybrid EHC-BGV approach with existing approaches.....	93
4.3.8	Comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with key size of 8 byte.....	95
4.3.9	Comparison of considered time for the implication of encryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with key size of 8 bytes.....	97
4.3.10	Comparison of considered time for the implication of decryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with key size of 8 bytes.....	100
4.4	Conclusion.....	102
4.5	Brief summary of this chapter .....	103
<b>Chapter-5</b>	.....	<b>104</b>
<b>Recommendations and Future work</b>	.....	<b>104</b>
5.1	Summary .....	104
5.2	Implications .....	105
5.3	Limitations and Future scope .....	106
<b>Bibliography</b>	.....	<b>107</b>
<b>Annexure II</b>	.....	<b>116</b>
	List of Publications .....	116
<b>Annexure III</b>	.....	<b>119</b>
	Publications Reprints.....	119

## List of Tables

Table 1.1	Various Homomorphic schemes	08
Table 1.2	Detailed analysis of FHE schemes (NEHE, BGV, AHEE, EHC)	16
Table 2.1	Computational complexity of various Homomorphic algorithms	38
Table 3.1	CP-ABE- Multi-tenancy parameters	45
Table 3.2	Algorithm for generating the token/keys	55
Table 3.2(a)	Primary data of generated key, salting values, general OTP for data access and data control, On-demand OTP for verification and validation, and instance of Data	59
Table 3.2(b)	Primary data of generated key, salting values, general OTP for data access and data control, On-demand OTP for verification and validation, and instance of Data	59
Table 3.3(a)	Conversion of the password Pooja@25 to binary bits. The matrix is an 8x8 64-bit formation	60
Table 3.3(b)	Conversion of salting key to binary bits	61
Table 3.4(a)	The 1st and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	61
Table 3.4(b)	The 1st and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	62
Table 3.4(c)	The 1st and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	62
Table 3.4(d)	The 1st and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	63
Table 3.4(e)	The 1st and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	63
Table 3.4(f)	The 1st and 3th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	63
Table 3.4(g)	The 1st and 2nd bit interchanging of each character in 8x8 matrix vector with 64-bit formation	64
Table 3.4(h)	The 2 <sup>nd</sup> and 8 <sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation	64
Table 3.4(i)	The 2 <sup>nd</sup> and 7 <sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation	65

Table 3.4(j)	The 2nd and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	65
Table 3.4(k)	The 2nd and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	66
Table 3.4(l)	The 2nd and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	66
Table 3.4(m)	The 2nd and 3rd bit interchanging of each character in 8x8 matrix vector with 64-bit formation	67
Table 3.4(n)	The 3rd and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	67
Table 3.4(o)	The 3rd and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	67
Table 3.4(p)	The 3rd and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	68
Table 3.4(q)	The 3rd and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	68
Table 3.4(r)	The 3rd and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	69
Table 3.4(s)	The 4th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	69
Table 3.4(t)	The 4th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	70
Table 3.4(u)	The 4th and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	70
Table 3.4(v)	The 4th and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	70
Table 3.4(w)	The 5th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	71
Table 3.4(x)	The 5th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	71
Table 3.4(y)	The 5th and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	72
Table 3.4(z)	The 6th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	72

Table 3.4(ab)	The 6th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	73
Table 3.4(ac)	The 7th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation	73
Table 3.5(a)	The 1st bit to 8th bit interchange with 57th bit to 64th bit in 8x8 matrix vector with 64-bit formation	74
Table 3.5(b)	The 1st to 8th bit interchange with 49th bit to 56th bit in 8x8 matrix vector with 64-bit formation	74
Table 4.1	The key generation time variations among EHC algorithm and Hybrid EHC-BGV	92
Table 4.2	The encryption time variations among EHC algorithm and Hybrid EHC-BGV	92
Table 4.3	The decryption time variations among EHC algorithm and Hybrid EHC-BGV	93
Table 4.4	Comparative analysis of proposed hybrid EHC-BGV approach with existing approaches with 8-byte key size	94
Table 4.5	Comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size	96
Table 4.6	Comparison of considered time for the implication of encryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with 8-byte key size	97
Table 4.7	Comparison of considered time for the implication of decryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with 8-byte key size	100

## List of Figures

Figure 1.1	Working of Homomorphic encryption algorithm.	06
Figure 1.2	Methodology of BGV homomorphic scheme	12
Figure 1.3	Methodology of EHC homomorphic scheme	13
Figure 1.4	Methodology of AHEE homomorphic scheme	14
Figure 1.5	Methodology of NEHE homomorphic scheme	15
Figure 3.1	Ciphertext Policy Attribute-based Encryption (CP-ABE) algorithm in multiple tenants	43
Figure 3.2	Enterprise-level multi-tenancy environment for data access control, authorization, and authentication	47
Figure 3.3	Hybrid approach using BGV and EHC Homomorphic algorithms	51
Figure 3.4	Steps for generating the token and the associated keys depending on the role of the user	53
Figure 4.1.1	Test case for creating multiple tenants in cloud architecture	81
Figure 4.1.2	Test case for uploading a file on cloud server on real-time	82
Figure 4.1.3	Test case for request for accessing a file	83
Figure 4.1.4	Test case for generating the token key which is received successfully on email id	83
Figure 4.1.5	Test case showing the mutated key after an attack	84
Figure 4.2(a)	Test case with EHC keys generation time and size	85
Figure 4.2(b)	Test case with token generation on email id	86
Figure 4.2(c)	Memory required by EHC	87
Figure 4.3	Implications of EHC algorithm based key generation time, encryption time and decryption time	88
Figure 4.4	Implications of Hybrid EHC-BGV Homomorphic based key generation time, encryption time and decryption time	90
Figure 4.5	Implications of Hybrid EHC-BGV Homomorphic based automated key filter scenario with bits-shuffling mechanism based on the 8-byte formation matrix by 56 parity rules key mutations	91

Figure 4.6	Test case with comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size	96
Figure 4.7	Test case with comparison of considered time for implication of encryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size	98
Figure 4.8	Bit wise comparison of encryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size	99
Figure 4.9	Comparison of considered time for implication of decryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size	101



## Annexure I

### List of Abbreviation

ABAC	Attribute based access control
ABAC	Attribute based access control
AES	Advance Encryption Standard
AES	Advance encryption standard
AHEE	Algebra Homomorphic Encryption scheme based on updated Elgamal
AKG	Authentication Key Generation
APG	Authentication Password Generation
API	Application interface
ASCII	American Standard Code for Information Interchange
Auth_V	Authentication Verification
AWS	Amazon Web Services
BGV	Brakerski-Gentry-Vaikuntanathan
BMIAE	Blockchain multi-instance iris authentication using ElGamal Homomorphic Encryption
CCA	Chosen Cipertext Attack
CP-ABE	Ciphertext policy attribute-based encryption
CPA	Chosen plaintext attack
CPS	Cyber physical system
CRT-RSA	Chinese remainder theorem with Rivest-Shamir-Adleman
CSWA	Cloud-based workflow scheduling
DAC	Discretionary Access Control
DDoS	Distributed denial of service
DES	Data Encryption Standard
DoS	Denial of service
ECC	Elliptic Curve Cryptography
EDoS	Economic Denial of Sustainability attack
EHC	Enhanced Homomorphic Cryptosystem
FHE	Fully Homomorphic Encryption
FTP	File Transfer Protocol
HE	Homomorphic encryption
HEA	Homomorphic encryption algorithm

HElib	Homomorphic Encryption library
HK	Hybrid Key
HTTPS	Hypertext transfer protocol secure
IaaS	Infrastructure-as-a-Service
IPFS	InterPlanetary File System
IND	Indistinguishable
IP	Internet address
JWT	JSON web token
KM	Key mutation
KP-ABE	Key policy attribute-based encryption
LPN	Learning parity with noise
LWE	Learning with errors
MAC	Mandatory Access Control
MITM	Man in middle attacks
NEHE	Non-interactive exponential Homomorphic encryption algorithm
NIST	National Institute of Standards and Technology
OOTP	On-demand one time password
OTP	One time password
P2P	Peer to peer
PaaS	Platform-as-a-Service
PHE	Partial Homomorphic Encryption
PHM	Prognostics and health management
PK	Public key
PWD	Password
PuK	Public key
R-LWE	Ring learning with errors
RBAC	Role based access control
RSA	Rivest-Shamir-Adleman
RT	Receiver Tenant
SaaS	Software-as-a-Service
SEAL	Simple Encrypted Arithmetic Library
SHA-2	Secure hashing algorithm
SK	Secret key
SLA	Service level agreement

SMEs	Small and medium enterprises
SOA	Service Oriented Architecture
SQL	Structured Query Language
ST	Sender Tenant
SWHE	Somewhat Homomorphic Encryption
TLS	Transport layer security
UIDAaaS	User Identity Authentication as a Service
V-Nodes	Verifier cloud nodes
VF	Virtual firewalls
VM	Virtual manager
V-Nodes	Verifier cloud nodes

## **Chapter-1**

### **Introduction and Overview of Cloud Computing**

The chapter presents an overview of cloud computing, multi-tenancy, virtualization, homomorphic encryption algorithm, access methods and hybrid approach used for the proposed model. The motivation of the research work is to reduce the research gaps, and the objectives of the proposed work are explained. This chapter introduces the concepts used in the proposed model and the organization of the thesis.

#### **1.1 Overview of Cloud Computing**

A standard definition by NIST for defining cloud computing is given as, “It is a model that requires minimal management effort where all the resources can be shared with multiple users in sharing mode but with customized approach. This customized approach is managed in a multi-tenant environment for enabling convenient use of the configurable computing resources” [1].

Cloud computing refers to providing various cloud- based services such as software, application, platform, storage and networking resources on a pay-per-use basis. It reduces the expense cost for setting up resources by allowing the sharing mode [2]. It eliminates the requirement of setting up of high-cost computing infrastructure. The ability to scale up and down the services according to the requirement of the client makes it flexible and easy to use. Cloud computing provides various hosting and storage services on the internet.

The cloud computing is playing a major role in the evaluation of technical life of mankind with the successful execution of various cloud services such as software, application, platform, storage, and networking resources on a pay-per-use basis. Cloud data security is always the topmost priority of Cloud Service Providers. The cyber-attacks on cloud data prove that there is a need for continuous research for finding various updated ways to protect the cloud data from unauthorized users.

### 1.1.1 Cloud Service Delivery Models

Choosing the right model to adapt in business is the most important task. An appropriate cloud model can be selected based on the fact, which layers client wants to control and which ought to be the duty of the cloud provider. Basically, there are three types of service delivery models:

- **Software-as-a-Service (SaaS)** – Clients use the provided application or software directly as on demand without installing on one's machine. Users do not have any control over network, storage, server, and application. Medium and small-scale business this cost-effective model. In simple words, SaaS looks like passing by public transport. Transports have assigned routes, and you share the ride with different passengers. Some of the examples are: Google Drive, Microsoft Office 365, Drop Box, Google voice assistant, Apple's Siri.
- **Platform-as-a-Service (PaaS)** - PaaS can be used to deploy, test, and organize different applications. A pre-configured OS is used, but reasonable amount of control over the rest of the configuration. Control can be managed by the cloud service provider or the client itself with a limited access. PaaS can be like taking a taxi. You do not drive a taxi yourself but asks driver to take you to the destination. Some examples are AWS, Google App Engine.
- **Infrastructure-as-a-Service (IaaS)** – Hardware resources like server, storage, data center are delivered in IaaS service model as a service. No control over infrastructure is provided to the client. OS can be customized according to the client's need. No hardware needs to be installed at client's premises. IaaS looks like leasing a car. When you lease a car, you pick the car of your choice and drive it wherever you wish, however you are not the owner of that car. If you need upgradation, simply lease another car. Some examples are Microsoft Azure, Amazon Elastic Compute Cloud, and Google Compute Engine.

### 1.1.2 Cloud Deployment Models

According to the organizational need of security and client's ability to manage the cloud, an appropriate cloud deployment model can be selected.

- **Private Cloud** – An infrastructure used by a single organization. Control may be managed by the third-party cloud provider or by the client itself. It can exist on-premises and off-premises and is the most expensive and most secured cloud deployment model.
- **Public Cloud** – It supports all the organizations who wish to use the cloud computing services. Completely owned by the cloud provider itself, exists commonly on premises of cloud vendor. Its inexpensive nature lets medium and small businesses prefer this cloud over others, though working on public cloud can be less secure.
- **Hybrid Cloud** – It is a combination of cloud types as per organization's requirement. It is the most preferred cloud deployment model, capacity and performance can be improved.
- **Community Cloud** – Multiple organizations part of a community sharing community resources are supported by this model.

## 1.2 Multi-Tenancy

A multi-tenant cloud is a cloud computing architecture which isolates the resources and client's data into tenants, hence providing better privacy and integrity of data. It allows client provider to share the resource among a number of tenants for better resource utilization at reduced cost. In simple words, a single instance of a resource is being shared by multiple users. A tenant is a group of users, organization or an individual user using the service. Each tenant is isolated from other tenants on account of privacy. No tenant can access other tenant's data or needs special access permission from cloud provider.

### 1.2.1 SaaS based Multi-Tenancy Challenges and Concerns

SaaS (Software as a service) application providers were based on multi-tenant architecture which is the most challenging testing area, since it involves testing multiple tenants' layers (Inner, Outer tenant's categories). Both, Functional and non-functional testing can be applied on a SaSE (Software as Service Enterprise) application.

- The isolation among the tenants is important to manage the work between them. The modifications done by one tenant should not affect the performance of another tenant. Multi-tenancy isolation should be carefully done in all the levels of the cloud infrastructure; it can be at functional or non-functional level.
- Tenant customization should be supported in run time in such a way that the changes made by one tenant do not affect other tenant's interface. The settings of the tenants are customized individually based on their role and access applicability. It is the most important requirement of any multi-tenant application. Updation in one tenant's interface should not halt the performance of others.
- An adaptive access control policy for each tenant must be maintained for collaboration among various tenants to protect each tenant's data.

### **1.3 Virtualization**

Virtualization simplifies management of resources by sharing and pooling resources for maximum utilization, a tenant will realise as if the resource is being used by him only privately. Virtualization technology allows cloud service providers to use one server as multiple virtual machines in a cost-effective manner. It is the key component of the cloud. It works on the principle of SOA (Service Oriented Architecture).

Hypervisor is the VM (Virtual machine) manager, responsible for managing resources between different operating systems running on a system at the same time. There is always a risk in working on VMs. If hacker gets access of hypervisor, every control is gone then. A virus or malware in any of the single computer system may spread to other systems and ultimately reach the hypervisor corrupting the whole virtualized environment.

Various types of virtualizations are Network virtualization, Server virtualization, Storage virtualization, Application virtualization, Presentation virtualization, Full virtualization.

## 1.4 Basic types of attacks

A cryptosystem is said to be indistinguishable if the attacker is unable to distinguish cipher text pairs for the message encrypted. So, it becomes difficult to access the file. An algorithm or technique is said to provide stronger security if it is indistinguishable under Chosen cipher text attack (IND-CCA) as it is hard to break.

- **CCA:** A chosen cipher text attack is an attack model which decrypts the chosen cipher texts to obtain the plaintext and the encryption key. The private key can be obtained by the decryption process.
- **CPA:** A chosen plaintext attack is an attack model which chooses random plaintexts to find the respective cipher texts and evaluate the encryption key. This type of attack is easy to break as compared to CCA.
- **KPA:** A known-plaintext attack can be easily implemented as the plaintext-ciphertext pairs are known to the intruder, the only requirement is mapping the keys in some order to obtain the encryption key. This attack can be easily breakable and hence is not a preferred choice of cryptanalysts.

## 1.5 Homomorphic Design

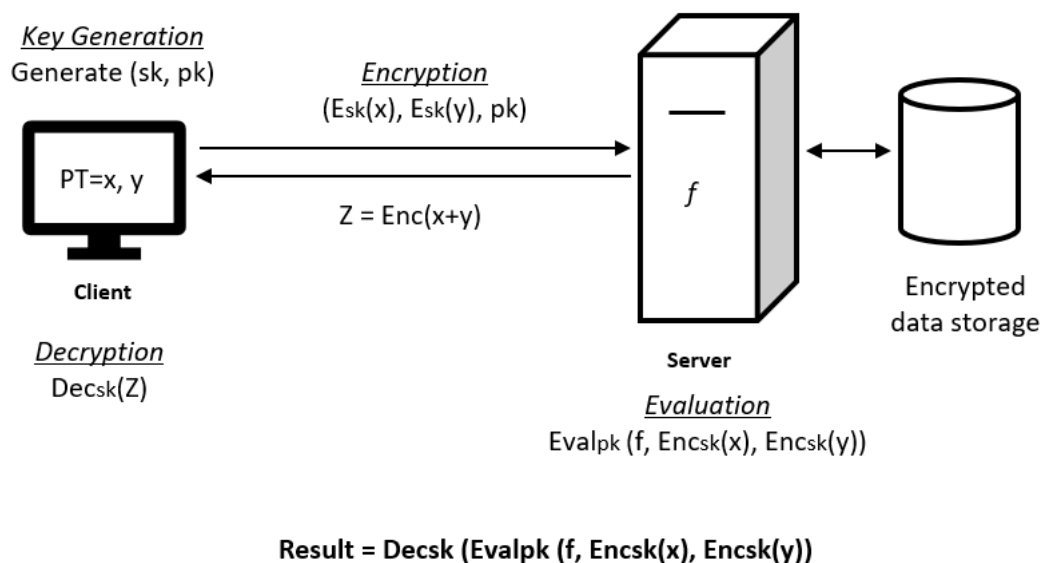
Homomorphic design allows cloud providers to compute the client's private data as requested by the client himself, without the need of decryption. Computations are done on encrypted data where cloud provider has no idea of the original data hereby protecting the data from unauthorized access.

HE (Homomorphic encryption) method is mainly used in Privacy protection, Data processing, Ciphertext retrieval. Traditionally, CSP (Cloud Service Provider) asks the client for decryption key, apply mathematical operation on original data and store the data over cloud in encrypted form again. This process violates the privacy feature in cloud. Hence is the need for Homomorphic Cryptosystem[3][4][5]. HE algorithm schemes are used to perform the operations and computations on the encrypted data in a secure manner. Decryption process is not required as is required traditionally to perform the evaluation of values stored on the cloud. This allows data integration, privacy, and protection.



For example, an organization may opt the third party for some computational work. Traditionally, the cloud providers or the organizations themselves put user's data at the risk, to operate and compute some results by taking help from third-party and the third-party asks for the decryption key, hence violating the privacy factor. But after the development of the homomorphic algorithm, third-party can work on the encrypted data only. Thus, it helps in maintaining the privacy of the user's data.

It is categorized into 3 types; PHE (Partial Homomorphic algorithm), SWE (Somewhat Homomorphic algorithm) and FHE (Fully Homomorphic algorithm). A PHE supports only one ciphertext operation at some instant of time that is, it can be either multiplication or addition operation. A SHE supports both the operations at the same time but with a limited number of computations. The FHE algorithm is considered as secure and flexible implementation as it allows an unlimited number of operations on the encrypted data. *Figure 1.1* shows the key generation process using HE algorithm. The secret key (sk) and public key (pk) are generated at the client's terminal and the data variables x and y are encrypted using the pk. The functions and operations are performed at the server terminal in the encrypted form and the computed values are sent back to the client where client can use sk to decrypt the ciphertext data. Hence, the computation takes place on the encrypted values which results in providing better security and protection to the cloud data.



*Figure 1.1 Working of Homomorphic encryption algorithm*

### 1.5.1 Functions of Homomorphic Encryption

HE consists of basically four functions, Generating the keys, Encrypting the plaintext, Decrypting the ciphertext, and Evaluation of operations [6][7][8].

**Function name-** Key generation (S)

This function generates the private and public keys by taking an input security parameter.

Input- Security parameter (S)

Output- Public key (PuK), Private keys (PK)

**Function name-** Encryption (PuK, PT)

This function encrypts the plain text using a public key and produces a resultant decrypted text that is, cipher text.

Input- Public key (PuK), Plaintext (PT)

Output- Ciphertext (CT)

**Function name-** Evaluation (PuK, Circuit, CT)

This function evaluates the cipher text on a circuit of inputs by applying the public key. The functioning is evaluated on the decrypted data, hence protecting the privacy of user's data. It returns another cipher text which is the encrypted resultant after computation.

Input- Public key (PuK), Circuit of  $t$  inputs, set of  $CT$  of  $t$  ciphertext  $CT_1$ ,  $CT_2$ , and so on.

Output- Ciphertext (CT)

**Function name-** Decryption (PK, CT)

This function returns the decrypted cipher text by applying the private key and returns the original plaintext.

Input- Private key (PK), ciphertext (CT)

Output- Plaintext (PT)

## 1.5.2 Homomorphic Encryption Schemes

Some of the homomorphic schemes are discussed in the tabular form in Table 1.1[8] [9]. The table contains the PHE algorithms like RSA, ElGamal, Paillier and FHE algorithms like BGV, EHC, AHEE, and NEHE.

Table 1.1 Various Homomorphic schemes

<b>Homo. Scheme</b>	<b>Homo. Type</b>	<b>Definition/Problem Taken</b>	<b>Nature of Data</b>	<b>Applications</b>
RSA Rivest-Shamir-Adleman [8]	Multiplicative Homo.	Asymmetric encryption, Security assumption on Integer Factorization Problem	Static	To secure internet banking
ElGamal [8][9]	Multiplicative Homo.	Based on the concept of Discrete Logarithms. Security assumption on Diffie-Hellman key exchange.	Static	Hybrid system
Paillier [8]	Additive Homo.	Based on Decisional composite residuosity assumption (DCRA)	Static	E-voting system
BGV [8]	Mixed Homo.	Asymmetric encryption based on encryption of bits. Works on integer values in polynomials vectors.	Static and Dynamic	Security of Integer Polynomials
EHC [8]	Mixed Homo.	EHC is the type of homomorphic encryption with its numerous applications based on real time. EHC is secure under CCA and is used for Encryption and Decryption operation.	Static	Transmission of messages securely, MANETS
NEHE [8]	Mixed Homo.	Encrypted polynomial and exponential functions are evaluated.	Static	Active Networks, E-Commerce

### 1.5.3 Properties of Homomorphic Encryption

HE has mainly two properties; Additive and Multiplicative Homomorphic Encryption [8] though other operations like subtraction and division can be applied using addition and multiplication operations.

#### 1.5.3.1 Additive Homomorphic Encryption

The condition for additive HE is,

$$E_k (PT1 \oplus PT2) = E_k (PT1) \oplus E_k (PT2)$$

**Paillier Cryptosystem (1999)**[8]: It follows additive HE property. The Paillier encryption algorithm belongs to PHE category as it supports only a limited number of addition computations on the cipher text.

Key generation:

Step 1: Calculate modulus value,  $n = pq$

Step 2:  $\lambda = \text{lcm}(p - 1, q - 1)$

Step 3:  $g \in \mathbb{Z}/n^2\mathbb{Z}$  such that  $n \nmid \text{ord}_n(g)$

Step 4: Public-key:  $(n, g)$ , secret key:  $\lambda, \mu$

Encryption of  $m$ :

Step 1:  $m \in \{0, 1 \dots n - 1\}$ , a message

Step 2:  $h \in \mathbb{Z}/n\mathbb{Z}$

Step 3:  $c = g^m h^n \text{ mod } n^2$ , a cipher text

Decryption of  $c$ :

$$m = L(c^\lambda \text{ mod } n^2) L(g^\lambda \text{ mod } n^2)^{-1} \text{ mod } n$$

The constant parameter,

$$L(g^\lambda \text{ mod } n^2)^{-1} \text{ mod } n \text{ or } L(g^\lambda \text{ mod } n^2)^{-1} \text{ mod } n \text{ where } g=1+n \text{ mod } n^2$$

can also be recomputed once for all.

### 1.5.3.2 Multiplicative Homomorphic Encryption

The condition for multiplicative HE is,

$$E_k (PT1 \otimes PT2) = E_k (PT1) \otimes E_k (PT2)$$

**RSA Cryptosystem (1978)[8]:** It follows multiplicative HE property. The RSA encryption algorithm belongs to PHE category as it supports only a limited number of multiplicative computations on the cipher text.

#### Key Generation

Step 1: Two large prime numbers  $p$  and  $q$  are selected for generating public and private keys.

Step 2: System modulus can be calculated as,

$$N = p \cdot q \text{ and } \phi(N) = (p-1)(q-1)$$

Step 3: Encryption key  $e$  is randomly selected where,  $1 < e < \phi(N)$ ,  $\gcd(e, \phi(N)) = 1$

Step 4: Public encryption key is defined by,

$$KU = \{e, N\}$$

Private decryption key:  $KR = \{d, p, q\}$  where  $d$  is the secret exponent which is used as a combination in the decryption key.

#### Encryption

Step 1: Receiver's public key,  $KU = \{e, N\}$

Step 2: Encrypted value is defined by,

$$C = M^e \text{ mod } N, \text{ where } 0 \leq M < N$$

#### Decryption

Step 1: Private key is input,  $KR = \{d, p, q\}$

Step 2: Decrypted value is computed by,

$$M = C^d \text{ mod } N$$

## 1.6 Homomorphic techniques for securing the data with data access control

The HE algorithm can be combined with the access control policies. These access policies are designed to provide customized access to the administrator or the other

users or tenants. Tokens are commonly used for the access control mechanism, which can be combined with multi-factor security parameters like login credentials, salting, key-mutation and so on [10][11]. Salting technique can be integrated with the login credentials like username and password. A one-time password can be added as an additional security protocol to protect the data on public cloud.

The FHE algorithm is considered as a strong encrypting algorithm than PHE and SHE algorithms. Different mechanisms can be followed for a single enterprise and multiple enterprises. A single enterprise may use limited security parameters because the users are known insiders and there is less possibility of the attackers and intruders getting access to the data. Tokens can be used for communicating securely in a single enterprise. The token is generated randomly and may expire after some fixed period of time and a new token is required to continue accessing the services. When data is transmitted from a tenant in one enterprise to another tenant in different enterprise, additional security protocols are required as data is transmitted publicly and risk of being attacked is more in this case. Relying on only tokens does not secure the data, a combination of public, private keys is to be used. Hybrid key is generated for communication between different enterprises.

The FHE schemes are considered as safe but it becomes difficult to protect the data and handle the access controls for privacy at the same time[12]. Hence, some other mechanism must be integrated with the FHE algorithm to secure the cloud data.

The research idea is to develop a more secure system with improved authentication and authorization techniques. In the proposed model, FHE scheme is integrated with the self-key mutation technique based on the access control mechanism.

## **1.7 Analysis of FHE Schemes (BGV, EHC, AHEE, NEHE)**

### **1.7.1 BGV (Brakerski-Gentry-Vaikuntanathan)**

A combination of public and private keys is generated and sent on cloud server, once encryption process completes. For performing computations, FHE does not require generating a decryption key. Refer *Figure 1.2* for its functioning. The data is sent to the cloud by encrypting at the user's end using the public key

and FHE operations are applied on the encrypted data and the encrypted computed values are returned to the user where user applies private key to decrypt the data and access the same[8][13].

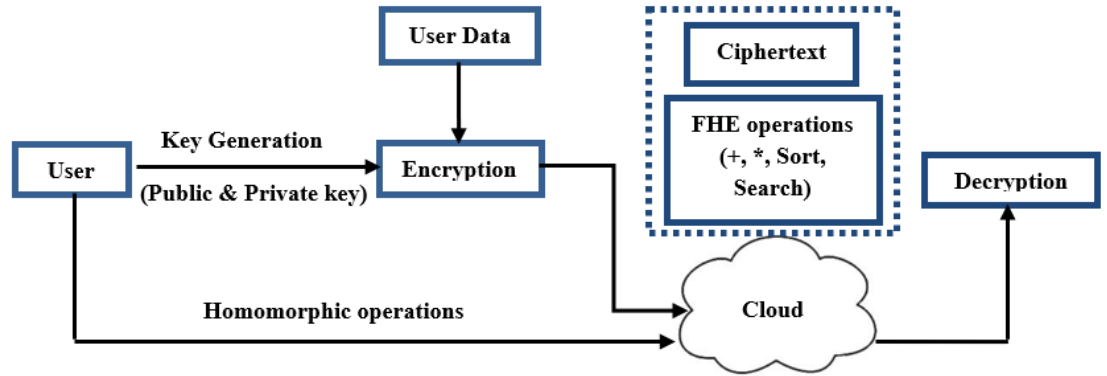


Figure 1.2 Methodology of BGV homomorphic scheme

### 1.7.2 EHC (Enhanced Homomorphic Cryptosystem)

Prime numbers which are difficult to factorize are generated in EHC algorithm. Two large numbers are randomly selected, out of which one is used as a PuK (public key). Three PK's (private key) are generated for encryption process. Total four keys are generated. Its working is shown in the flowchart, refer *Figure 1.3*. Two large prime numbers are chosen such that it increases the complexity of guessing these numbers [8]. Modulus is calculated by multiplying these prime numbers and the result is shared as a private key and a random number or token is generated. Out of the chosen two prime numbers, one is taken as a public key while the other number is shared as a secret key or private key. Then the encryption process is performed using the formula  $Y = (X + r*pq) \pmod{m}$  where X is the message to be encrypted, r is the random number which is kept secret and p and q are the randomly generated prime numbers. The encrypted message is shared to the receiver's node where the ciphertext is decrypted to generate the original message, X.

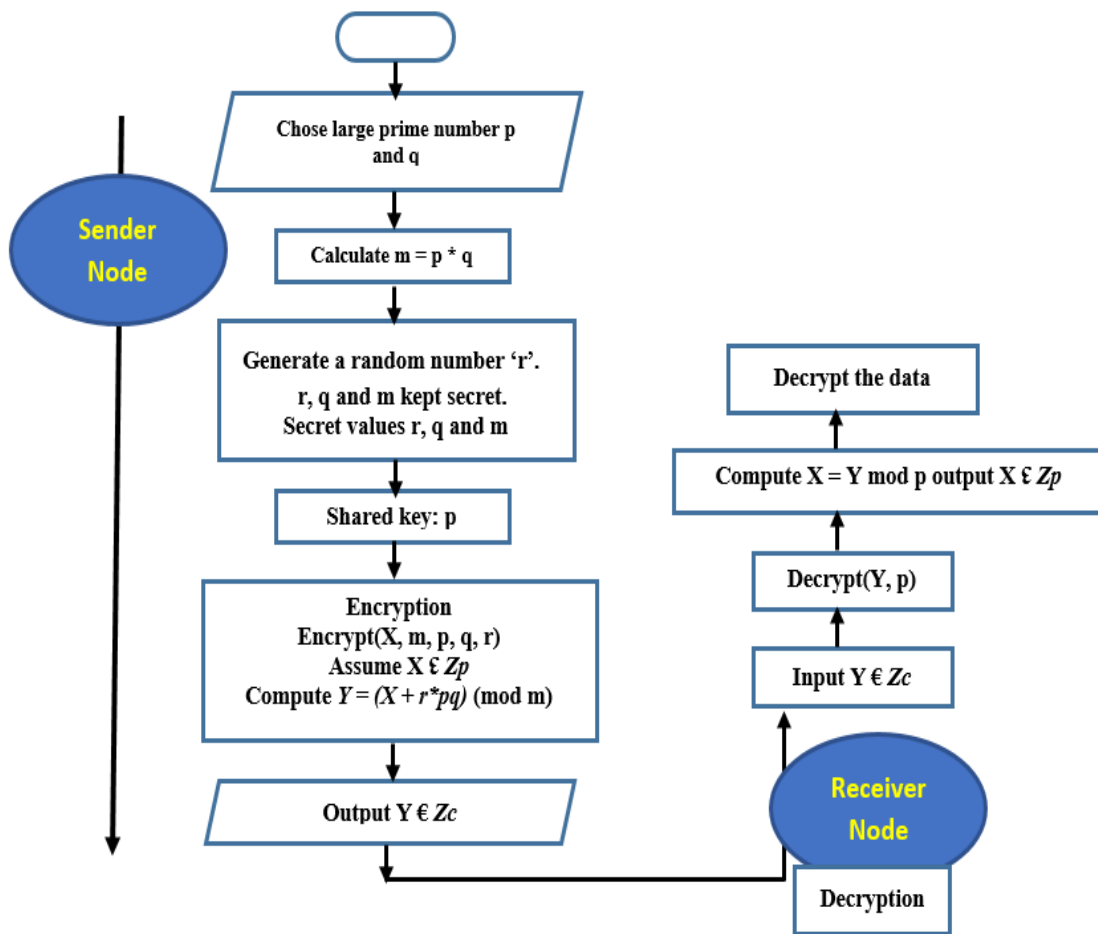


Figure 1.3 Methodology of EHC homomorphic scheme

### 1.7.3 AHHE (Algebra Homomorphic Encryption scheme based on updated Elgamal)

In this scheme, PuK is generated by multiplying the two very large prime numbers. PuK is used in the decryption process. A randomly selected number for encryption is used as another key and this step takes two sub-steps. Refer *Figure 1.4* for its functioning. The encryption process is divided into two steps, first uses a random number  $r$  and apply the encryption process and second one generates another random number  $k$  and again applies the encryption process [8]. The encrypted message is sent to the receiver's end where the private keys and the random number are applied on the ciphertext or encrypted message to get the original message  $M$ .



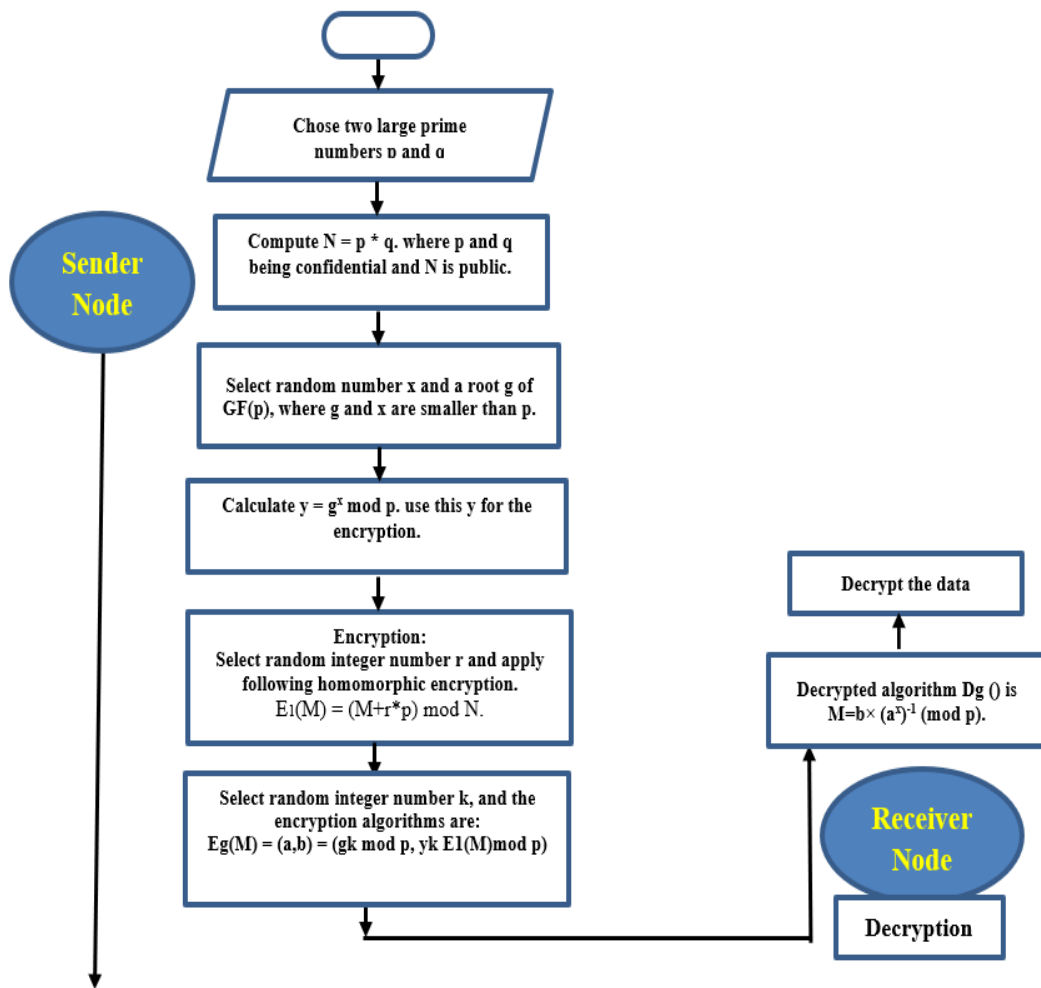


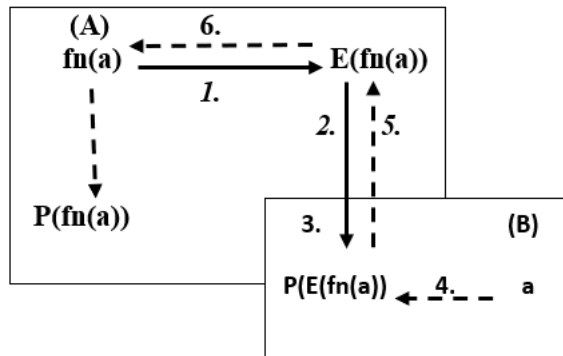
Figure 1.4 Methodology of AHEE homomorphic scheme

#### 1.7.4 NEHE (Non-interactive exponential Homomorphic encryption algorithm)

NEHE is built on RSA algorithm. In this scheme [8], four keys are generated; one PuK and three PK's. As shown in the *figure 1.5* for the key generation, three large prime numbers are selected. Encryption key is based on difficulty of factoring the number and is randomly chosen. The receiver must have the value of product of the three prime numbers that are selected and its input value x when it evaluates the encrypted function  $E(fn(a))$ .

Suppose the source host 'A' needs to compute a function 'fn' and the destination host 'B' can compute that function using the algorithm on some input value 'a'.

The condition in this scenario is that 'A' does not want to reveal any detail of the function 'fn' to 'B' and no communication can take place between 'A' and 'B' directly during the computation process. NEHE algorithm is fulfilling all the conditions in this scenario.



*Figure 1.5 Methodology of NEHE homomorphic scheme*

1. The encryption process for encrypting the function 'fn' is done at the source host end.
2. The source host 'A' will create a program  $P(E(fn))$  to implement  $E(fn)$ .
3. The program  $P(E(fn))$  is sent by host 'A' to host 'B'.
4. Host 'B' will execute  $P(E(fn))$  at input 'a'.
5. Host 'B' will send  $P(E(fn))(a)$  to 'A'.
6. Finally, host 'A' will decrypt  $P(E(fn))(a)$  and obtain  $fn(a)$ .

### 1.7.5 Detailed Analysis of various FHE Schemes (NEHE, BGV, AHEE, EHC)

Table 1.2 Detailed analysis of FHE schemes (NEHE, BGV, AHEE, EHC) [13][14][8]

<b>Name of Scheme</b>	<b>NEHE</b>	<b>BGV</b>	<b>AHEE</b>	<b>EHC</b>
<b>Functionalities</b>	The coefficients of a polynomial and Exponents are encrypted at the same time.	The bits can be encrypted using uneven encryption approach and noise can be reduced using the modulus switching technique.	Updated El-Gamal algorithm is used for encryption and decryption process[15].	Secure information exchange is the basis for encrypting and decrypting the data on cloud.
<b>Methodology</b>	RSA algorithm is the basic of the NEHE scheme. Only public key and the input value is required to evaluate the non-interactive exponential function[14].	Ciphertext rescaling and switching of keys are done to improve the security. In addition to the basic operations, level shifting operations are implemented [3].	A two-step encryption process is followed where two keys are generated by the user which can be used for decryption as well[15].	Secret key or private key is generated randomly for every encryption process resulting in a more secured environment as same plaintext will have different ciphertexts [16].
<b>Number of Keys</b>	One PuK and three PK are generated.	Two keys are generated	Two keys are generated[3] [15]	One PuK and three PK are generated[16][17]
<b>Advantages</b>	The structure of the polynomial function can be hidden from the attackers with NEHE scheme.	Ciphertext is managed by reducing the noise. No bootstrapping is	AHEE scheme provides better security and data efficiency and	EHC requires low power consumption and decryption time is comparatively

	The operations in NEHE are exponential, addition, multiplication and mixed Homomorphism[14].	required[3][18]. It supports the static and dynamic data both. The implementation of BGV is easily available in the libraries.	correctness is achieved[15]. It supports both the static and dynamic data.	less[17]. It is secure under CCA[16].
<b>Lagging Issues</b>	Dynamic data is not supported.	The storage management in BGV is a complex task. Hence, it is not suitable for real world applications [11].	It is not indistinguishable under CCA[15]; it is only IND-CPA secure.	The encryption framework must follow a format and structure to match the decoded information[8][19]. Only static data is supported.
<b>Applications</b>	Active networks, E-commerce mobile agents[3][8]	Overseeing integer polynomials and vectors based on LWE[8].	Suitable for multi-party computations applications, wireless networks and mobile cipher[8].	Mostly suitable for real-time applications; Mobile Ad-Hoc message transmission networks[8][17].

### 1.7.6 Selection criteria of BGV and EHC FHE schemes

#### BGV-

- Work on static and dynamic data both
- IND-CPA secure
- Overhead of storage
- 2 keys are generated

- Not suitable for real time problems.

#### **EHC-**

- Works on static data only
- IND-CCA secure
- Requires low power and least storage space
- 1 PuK and 3 PK are generated.
- Used mainly for real-time applications.

Based on the properties identified, it is found that a combination of BGV and EHC homomorphic schemes can be compatible for the proposed research topic. A combination of these schemes overcomes the disadvantages of both the algorithms resulting in a more secure algorithm.

### **1.8 EHC, BGV and Hybrid EHC-BGV homomorphic algorithm**

Factors to consider while choosing a scheme are computational performance, support for different programming languages, robustness and quality of available implementations, and the relative ease-of-use.

EHC and BGV fully homomorphic encryption algorithms are considered in the proposed research model because of their compatibility and suitability. In the proposed research, a hybrid EHC-BGV approach is generated that uses EHC to generate the tokens, public and private keys, and BGV algorithm is used to generate the hybrid key.

EHC (Enhanced homomorphic algorithm) is suitable for real-time applications but it can work on static data only. It generates 4 keys; 1 PuK and 3 PK. EHC does not require much storage for its implementation and power requirement is also less compared to other schemes. It is IND-CCA secure [16].

BGV (Brakerski, Gentry and Vaikuntanathan) can work on both static and dynamic data. Storage overhead in BGV does not allow to work with real-time applications. Only two keys are generated; 1 public and 1 private key. BGV is indistinguishable under CPA (chosen plaintext attack)[13].

A combination of BGV and EHC schemes overcomes the disadvantages of both algorithms resulting in a more secure hybrid algorithm. Since BGV requires extra space, EHC overcomes this by consuming less storage and EHC supports only static data while BGV overcomes this as it supports static and dynamic data both [3]. Hence, this hybrid approach is found to be more secure.

## **1.9 Brief summary of this chapter**

This chapter introduces the basic idea of the proposed research. The concept and terms relevant to the research work like homomorphic encryption algorithm, virtualization, cloud models etc are discussed in this chapter.

## **Chapter-2**

### **Literature Review**

This chapter includes some of the studies and methodologies by various authors in the previous years, most researchers have incorporated techniques for security and privacy among the multi-tenants of single-multi enterprises.

#### **2.1 Analysis of literature**

##### **Tenant isolation:**

Y. wang et. al. [20] analyzed that a performance technique for tenant isolation can be used as sharing resources results in the competition for service performance. The technique analyzed by authors includes an SLA (Service level agreement)-oriented multi-tenant hybrid scheme. For isolated performance among tenants the container technology is used. The proposed study results in an improved and cost-effective version. The future scope of this study is to include more factors and parameters based on security among tenants. The authors of this paper Sastry and Basu [21] ensures secrecy through multiple databases and multi instances situated in different machines. Data isolation and Application isolation in Multi tenancy are the issues considered in this paper. Eucalyptus tool is used to implement the algorithm developed by modifying RSA algorithm as changing the decryption process and the key generation. Double encryption method is used by user and CSP (Cloud Service Provider) both. Algorithm developed in this paper ensures secrecy as attacker is not able to gain access to the location of the data or the key used for encrypting at CSP, not even using brute force attack. The proposed method is expensive as a greater number of DBMS instances must be created to ensure secrecy.

##### **Multitenancy and security issues in cloud computing:**

Jumagaliyev et. al. [22] describes a method to support abstract Multi-tenant data architectural model for different types of cloud storage. To implement multi-tenancy over these different types of storage, data layer code is generated. Main focus is on Platform-as-a-service. Multi-tenancy at data layer using manual coding is time

consuming and prone to errors. A domain-specific modelling language, CadaML is introduced. It transforms model to source code automatically for different cloud storage data types namely, relational database, non-relational database and blob/object storage. CadaML is a graphical language, hence easy to learn. It is not required to create own multi-tenant-safe implementation. CadaML manages storage types and there is no need for testing and evaluation phase as, any error if exists is handled at the model level. Future scope is to check the performance of the proposed method on real cloud applications in terms of code reliability and overhead of developing the code. Kanade and Manza [23] presents a survey on SAAS (Software-as-a-Service) Multi-tenancy applications. A well-defined method is required at all the layers namely; Server layer, Network layer, Application layer and storage layer. Current approaches are infeasible in accessing the control to multi-tenancy requirements as individual user IDs are used in most of the cases. There is always a risk of unauthorized access of data in multi-tenancy. State-of-the-art survey of multi-tenancy is done, which provides better understanding of the design challenges and concepts of multi-tenancy. This paper helps researchers to find out various aspects of multi-tenancy for future work. Future research can be made on the basis of present design issues in multi-tenancy. Hugo et. al. [24] reviewed the current literature through a Systematic Mapping Study, to evaluate main challenges in SaaS Multi-tenant environment. Multi-tenant architecture implementation challenges and future research opportunities mainly focused on SaaS, tenant customization in isolation and lack of standardization are some of the issues studied in this paper. A Systematic Mapping Study involving three phases: planning, conducting and reporting are conducted which selects study of 89 primary papers and is able to answer two research questions related to multi-tenant environment. Comparison of study/ evaluation done in this paper to industrial projects of cloud can be made in future research which may implicate new hypothesis for testing SaaS applications. Sahu and Pal [25], index structures were used for better performance and searching scheme. Encryption security schemes and data search schemes are focused in this paper. The cloud computing auditing protocol is composed by combining seven algorithms: SSetup, EUpdate, VESK, DESK, AuthGen, Proof- Gen, Proof Verify and Check Proxy TPA. Two index structures supporting efficient product vector retrieval and a hash value AVL tree are designed which supports feature-vector-based and identifier-based



product search. Future focus is on the protection of commercially confidential cloud data along with the ability to search data. Ramkumar and Gunasekaran [26] proposed two algorithms for improving the security and scheduling issues in cloud computing. They focused on security and scheduling issues. In the proposed model, Collocate FCFS of supremacy elements scheduling algorithm can be used for better utilization of resources. And for security, Crisscross AES (Advance encryption standard) can be used in grid manner. Comparing FCFS and collocate FCFS priority algorithm, the latter one is found to have least waiting time. On the other hand, analysis between AES and crisscross algorithm results in better throughput in case of crisscross algorithm. The throughput and WT can be analyzed for increased complexity of proposed algorithms. Further investigation can be done to implement parallel computing and efficacy of other cryptographic algorithms. Chowdhury [27] concluded that the security issues being faced by the internet are the same with the cloud computing hence, needs to be accessed securely. Major security challenges were discussed in this paper like remote access mechanism and SQL injection, DoS (Denial of service), DDoS (Distributed denial of service), man in middle attacks. Provides better way to implement access control mechanism, auditing and monitoring by reducing the possibility of occurrence of risks. According to interview report on major security risk, it is found that information security is the major risk area followed by the disaster recovery. Data transmission is equally important as data storage. Further exploration of security challenges should be concerned related to data transmission, storage security, application security and third-party related issues. Dahiya and Rani [28] proposed a **multilevel authentication technique** to enhance the data security in the cloud systems. DES (Data Encryption Standard) and AES (Advance Encryption Standard) along with RSA (Rivest-Shamir-Adleman) algorithm is used to protect from unauthorized access. **Any unauthorized access may lead to blocking IP address of that user after three failed attempts.** A combination of DES, AES and RSA algorithm will strengthen the security of the cloud. Transferring data directly to cloud for computing the operations from storage cloud may lead to data integrity issue. Nadeem [29] investigates cloud architecture and various protocols used for any weaknesses. Weaknesses in cloud architecture and security protocols are discussed in this paper. Surveys the weaknesses in present security protocols, cloud architecture, application software and the cryptosystem. Challenges

related to cloud security were identified and counter measures to resolve those issues. Further investigation can be done to resolve the issues discussed. Bhadauria et. al. [30] identifies various security issues in cloud computing and their possible solutions. If one of the servers used in SaaS is damaged then control of data is lost. In PaaS, outage problem in case of congestion is there. In IaaS, a governance framework is required to implement virtualization technology. Various security threats discussed based on different levels; Application-level security, Network level security and Basic Security. Proper encryption decryption techniques can be used to secure the data over cloud from upcoming threats for future research. Nazir [31] highlights basic building blocks of cloud, its architecture and related research challenges. Cloud computing architecture, building blocks and entities were introduced. This paper provides better understanding of the design issues or challenges of cloud computing which may be used by other researchers. Further research can be done on the design challenges discussed. Varsha et. al. [32] addresses various issues based on literature review on Multi-tenancy issue. Literature review is used for gathering information related to multi-tenancy. Basic framework of multi-tenancy is introduced which can be used for further research. A more secured framework can be designed to satisfy the security issues related to multi-tenancy. Dharani et. al [33] used Logistic regression classification algorithm to achieve the highest accuracies with low false positive rate. Logistic regression classification algorithm and Weka data mining tool are the techniques used. Four measures were included: Accuracy, Precision, Recall and False Positive Rate in the research. It is found that the experimental results using LogitBoost classifier achieves highest accuracy of 97.90% while false positive rate of 0.009 is found. Furthermore, research can be done to improve the security on intrusion model. Kumar et. al. [34] used proper scheduling algorithms to distribute load among various tenants in virtualization, keeping in mind the security concerns. This paper uses scheduling algorithms to assign the resource to the request made by the user. Some algorithms used were round robin, weighted round robin along with faster response time. A hyper-safe program known as passable memory lock down can be the solution to the hypervisor security issue. Under this scheme, no new code can be introduced by any user other than the hypervisor. Further work can be done to evaluate various issues related to virtualization. Su et. al. [35] proposed a model to generate user test cases in penetration testing of SQL injection

attack model based on transmission channel. The proposed model for SQL injection detection can describe all types of SQL injection types and reflects the rules and characteristics of different attack methods in SQL injection efficiently. Also, it can reduce the false alarms and omissions. Experimental analysis is done using sqlmap. Further research can be done to identify such vulnerabilities and their proposed solutions. Rimal [36] introduced a workflow scheduling system with four layers for resource management in Multi-tenant environment. A novel CWSA (Cloud-based workflow scheduling) policy is employed with proof-of-concept experiments. CWSA policy minimizes the execution cost of the workflows, utilizing idle resources. CWSA outperforms as compared with other scheduling policies like FCFS, Backfilling, and Minimum Completion Time (MCT) in terms of performance. Inherent heterogeneity and resource isolation may increase the complexity in managing resources in multi-tenancy. Sakthipriya et. al. [37] explores various cloud security issues such as: confidentiality, authenticity, management of keys, data splitting. Focus on designing a better cloud environment. The better understanding of the challenges can pave the way for future research. Research must be done in future on these security concerns and finding their improved solutions. End-to-end security should be evaluated for the cloud users. Sqalli et. al. [38] proposed a system that will help to reduce the latency time while executing the services, since unauthorized request is denied of accessing any cloud service hence providing better EDoS (Economic Denial of Sustainability attack) shield protection mechanism. EDoS shield architecture is created, with main components as VF (Virtual firewalls) and V-Nodes (Verifier cloud nodes). The location hiding problem is solved in this approach as in overlay-based approaches. Future work should be done to deal with IP spoofing attacks and enhancing the proposed architecture. Since, the decision to forward or dropping a packet completely relies on source IP address found in blacklist or whitelist.

**Genetic algorithm- key mutation concept:**

Alkharji et. al. [39] analysed a method for generating random keys for the fully homomorphic encryption algorithm scheme. The authors focussed on providing strong encryption method using FHE scheme which is integrated with the genetic algorithm. Genetic algorithm is used to generate public and private keys randomly. The keys are

generated by analysing the population size, cross-over and mutation techniques. It is found that using Genetic algorithm, the security of FHE scheme can be enhanced. Arshad et. al. [40] proposed an improved genetic algorithm to generate the random keys for encryption. The genetic algorithm operators are modified by adding local intelligence which contains the local information and a random bit generated in each iteration. For encryption process, the source data is converted to binary bits and after the encryption process, this binary data form is converted to ciphertext again. The proposed model is considered as 80% more efficient than the conventional genetic algorithm. In future, the proposed algorithm can be analysed with increased complexity and large volume of data. Other encryption schemes must be evaluated with the proposed scheme. Kalaiselvi et. al. [41] proposed symmetric key homomorphic algorithm which is based on genetic algorithm for generating the random keys. Since the keys are generated randomly for each iteration, a very strong cryptosystem is developed. The proposed model is secure CCA. Future scope is to check the proposed model with public key cryptosystems.

#### **Authentication and verification models:**

Veeraragavan [1] proposed UIDAAA service to secure the cloud which includes three algorithms: APG (Authentication Password Generation), AKG (Authentication Key Generation) and Auth\_V (Authentication Verification). The proposed model will solve user authentication design and deployment issues. Windows azure and ASP.NET language is used to design and deploy the proposed UIDAAA authentication algorithm. Security level of existing and proposed authentication mechanism can be measured by Hackman tool using DoS and MITM (Man in the middle) attacks. Outcome of the comparison shows that User authentication mechanism provides better security than the existing methods. Authentication Key is only a single character which takes very less space in memory. Time taken in authentication process is less comparatively. The performance of the proposed algorithm can be measured for attacks other than DoS and MITM attacks as a future scope. O. Ethelbert et. al. (2017) [10] proposed a method for improving the authentication process based on JSON model. The HTTPS/TLS transmission is suggested to use for a dual authentication and authorization process. It is found that the peak time of access demands can be handled without any latency as

the stateless and compact feature of JWT (JSON web token), GAR (Granted Access Rights), UAD (User, App or Device) are highly portable. **The dual authentication process is not sufficient for the future threats and some additional authentication modes are required to secure the data from unwanted breaches.** Al-Attab and Fadewar [11] introduced a new device for authentication of the user in cloud system. This device is a USB (Universal serial bus) which generates random key, every time user needs to login as second level of security. Dual identification technique used for second level of security. A dual identification scheme improves the security of the users. User's data can be prevented from phishing attacks and brute force attacks. Future work can be done for designing such device with improved security, and from view point of user, CSP (Cloud service provider) and the browser. Awasthi [42] introduced a novel and more secure verification arrangement that depends on **E-mail based OTP**, which is secured by Java Mail API and hash key has been proposed. Identity access management, character-based verification and characteristic based confirmation are the significant security issues discussed in this paper. **A single one-time token key is generated for identity management of the client. Every time a new token key is generated, the previous key or watchword is destroyed from the cloud framework.** The proposed model is secured against session hijacking attack and brute force attack. Session time out mechanism is available. Future work can be done in administering key process providing better client validation. Also, research can be done in minimizing the execution time. Gauravaram [43] analyzed that hash function can be used to protect from birthday and dictionary attacks, to the combination of salt and password. An offline birthday forgery attack presented a contradictory belief that birthday attacks can be complicated using hash values with prepended salts computed over the passwords. It was observed that offline birthday forgery attacks cannot be prevented using the hash function. Future scope is to develop a more secure system to prevent from these attacks. Rakesh et. al [44] designed an effective scheme that guarantees the **data storage correctness**. For distributed verification of erasure-coded data, Homomorphic token is used. This development reduces the storage and communication overhead. **Future scope is to find a scheme where we can achieve both the data storage correctness and public verifiability of dynamic data.**

### **HEA (Homomorphic Encryption Algorithm):**

Rivis and Zhu [45] suggested that the key management challenges can be managed by introducing Homomorphic encryption and re-encryption together, as a result for every new recipient, encrypted data is re-encrypted without having to decrypt it. HEA is used to compute encrypted data without decrypting it. Since data operations cannot be applied on encrypted cloud data, one has to first decrypt it which leads to privacy issue. The authors Ahmad and Khandekar [4] developed RSA and Paillier algorithm for HE using proxy-Re-encryption algorithm. The proposed algorithm can be used to generate random key cipher text every time. If in case attacker gets the key then only that plaintext will get access, all other plaintexts are safe. Proxy Re-encryption method can be tested with other HE schemes and size of the key can be reduced for efficiency. Chen and Zhao [5] concluded that fully HEA leads to better security in the cloud. DHCV and CAFED algorithms are used to protect the cloud data. In DHCV, attacker has to attempt 2512 times to get access to the cloud data, which is technically near to impossible. In CAFED, Data access and data processing are kept isolated to secure private data. These algorithms can be further investigated for better security so as to follow fully HEA scheme. Prasad and Kumanan [13] introduced Enhanced BGV technique by modifying the classic fully homomorphic encryption scheme BGV. A new technique for sorting was proposed that sorts the encrypted data without requiring a decryption key. A number of experiments are executed using the eucalyptus tool. The authors Parmar et. al. [8] discussed four fully homomorphic encryption schemes BGV, EHC, AHEE, NEHE in this paper. The number of keys used in each scheme, pros and cons and implementation details are presented. The application area of these four schemes and their suitability is discussed in this paper. Ayman Alharbi et. al. [46] introduced a literature survey related to the homomorphic encryption algorithm. The survey aims to reduce the gap between the algorithm and its applications in terms of providing security. The homomorphic applications like vehicle communication, electronic voting, cloud computing, Blockchain, and signal processing are discussed in the paper. The homomorphic algorithm is considered as providing a high level of security by allowing the operations on encrypting data. In the future, a complete systematic view of the homomorphic algorithm can be analyzed. The authors Kang et. al. [47] showcase how

HE algorithm can be used to outsource PHM (Prognostics and Health Management) services securely and privately in SMEs (Small and Medium Enterprises). A two-party collaboration H-FFT-C is designed using HE algorithm and two sub-algorithms are developed from H-FFT-C as H-FFT and H-C. The first one extracts information related to the frequency of the manufacturing machines and the second one is used a comparison operator that computes values and gives result in milliseconds. The combination of these two sub-algorithms that is H-FFT-C algorithm outsources PHM services efficiently. The result after running the proposed methodology on a fiber extrusion device shows that the server successfully implements the computational part and delivers the machine health report. Future scope is to extent this research for multiple SMEs for a multi-party collaborative scenario. Min et. al. [48] proposes a feasible solution for the security challenges in a CPS (Cyber Physical System). It combines the FHE technique with the CPS which allows ciphertext operations on the encrypted data without the need of decryption. The current homomorphic encryption allows operations on a limited data type, the authors of this paper suggest using a parallel approach on that supports floating-point numbers. Group wise ciphertext operations are performed parallelly to enhance the security using out-of-order ciphertext operations. The proposed algorithm is implemented in MapReduce platform, resulting in better speed with big files processing. Costache et. al. [49] investigated BGV and FV schemes in this paper. A comparison is made between these two schemes based on their noise behaviour and the method used for analysing the noise will upper bound the growth loosely. The two libraries used for implementation are HELib and SEAL. The authors suggested to choose a particular scheme based on some factors. The gap between the noise evaluated in the heuristic upper bounds and the result from practical experiments can be reduced as a future scope Chen et. al. [14] discussed the FHE scheme NEHE. The exponents and coefficients of polynomials are encrypted using NEHE homomorphic scheme. Algebraic HE with RSA used in NEHE provides better security as the structure of the function is kept hidden and no decryption key is required. The homomorphic schemes El-Gamal and EHC are analyzed by the authors Al-Mashhadi and Khalf [16]. These two algorithms were used to build three HE cryptosystems based on the block pixel position method to securely transmit the digital images on the server. EEOE (El-Gamal-EHC based on Odd and Even block index),

EEBPT (El-Gamal-EHC based on block position in lower Triangle) and EEZSC (El-Gamal-EHC based on Zigzag Scan and Counter) are the three cryptosystems developed. The proposed methodology is considered as safe and takes less execution time. Fahina et. al. [15] secures the data in cloud using FHE scheme AHEE which is based on the updated El-Gamal algorithm. It is DSS a modified form which randomly generates the key for every  $E1()$ . The proposed model is indistinguishable under chosen ciphertext attack. Rao et. al. [17] analyzed EHC scheme for MANETs in this paper. In the proposed model, encryption of messages over MANETs is done using EHC algorithm. The messages are divided into groups with message id. Cipher values added with the message id will result in the original decrypted message. To get access to these messages on MANETs, an attacker requires a decrypted messages from each group, which is a complex task. Hence, security of the proposed model is considered as high as compared to other algorithms in the paper. **Mixed multiplicative HE, EHC and El-Gamal encryption schemes are compared and it is observed that processing time in EHC is very less.** N. Sammeta and L.Parthiban [50] analyzed in their paper that the FHE AHEE scheme is used to process the medical data privately and in a secure manner in the hospitals. By checking the patient history, an upcoming health issue that may or may not occur can be alarmed so that the patient can take appropriate medications in advance to avoid future health issues. The medical records can be kept secret from the cloud administrators using the proposed model. The time taken for encryption and decryption process is comparatively less. The paper can be further implemented by adding bio-metrics like patient's fingerprint, pupil tracker and so on. R Kanagavalli and Vagdevi S [3] compared the BGV HE scheme with RSA and AHEE homomorphic schemes based on byte level automorphism on BGV. The parameters used for comparison are time taken for encryption process, size of the ciphertext generated and time taken in decryption process. The testing is done on different file sizes. **It was found after comparison that the encryption and decryption time taken by BGV is less compared to RSA and AHEE. But BGV takes large storage space and for real-time applications managing memory becomes a tedious task.** Future scope is to analyze methods that minimizes memory requirement and researchers can validate the proposed model for variable block size. The authors Sadeghikhorami and Safavi [51] presented a secure estimation strategy by considering Kalman filtering technique



integrated with Paillier's algorithm. Paillier's algorithm can only compute integer values hence some conversion was required but the proposed methodology can handle quantized data effectively. Earlier methodologies encrypt data before sending to the network and a local estimator calculates the state estimation hence, reformulation of estimator was required to receive the encrypted data and to produce the encrypted output. The proposed approach reformulated the estimator for quantized data. Hence, the proposed strategy protects the confidentiality of data by protecting against eavesdropping attack. Awadallah et.al. [52] proposed a model for validation and verification of the HE computations for finite integer values. **A HE algorithm alone is not sufficient for providing the data integrity as the computations may be swapped.** The authors of this paper have introduced a verifiable scheme to validate the HE computations. The proposed scheme adds overhead of managing increased storage requirement and computations which is completely acceptable according to the authors. D. Bhatia and M. Dave [53] concluded in their paper that implementing FHE schemes is a complex process and the **applications developed using FHE still lacks in security.** The authors proposed "Privacy Homomorphism" that permits multiple computations securely. Various algorithms are compared in this paper. It is found that ECC (Elliptic Curve Cryptography) gives better results and is suitable for sensitive FHE applications. T. Oladunni and S. Sharma [9] focus on using HE algorithm for protecting the cloud data. A HE algorithm is categorized depending on the operations into three types. A PHE algorithm performs only one operation at a time that is, it can be addition or multiplication operation, a SWHE algorithm allows more than one operation like addition and multiplication operation at the same time but with the limited number of operations. While a FHE algorithm allows unlimited ciphertext operations. RSA, Paillier, and Elgamal algorithms are PHE algorithms, R-LWE and LWE are SWHE algorithms as these are noise-based, and EHC, BGV, AHEE, NEHE are FHE algorithms. The computations done on encrypted using FHE algorithm data results in maintaining the integrity, privacy and security of cloud data. **Integrating Blockchain technology with cloud computing:**

Shangping Wang et. al. [54] proposed a method to improve access control policies. Ethereum Blockchain and CP-ABE (Ciphertext policy attribute-based encryption) are used

for this purpose. Ethereum Blockchain is deployed on Linux/Unix operating system while Windows 10 is used for the implementation process. A smart contract is used to store the ciphertexts. Cloud security is improved by decrypting in a valid access period. Accessing the cost of the files has decreased, making function tracing easier. To improve data integrity, decentralized technologies can be introduced. Adamu Sani Yahaya et. al. [55] locates a potential supplier for demander in Electric vehicles by proposing a privacy preserving algorithm. The supplier can be searched using a P2P communication approach and the implementation takes place using a PHE. Blockchain verifies the energy transmission process. The proposed model is found to be faster than BMNN (Bichromatic Mutual Nearest Neighbor) algorithm. Performance of the proposed model can be optimised by implementing on the methodology hardware. Ch. V. N. U. Bharathi Murthy et. al. [56] discusses the problems with the cloud and suggests to use Blockchain with the cloud. An integrated architecture is developed by surveying the Blockchain on a scalable cloud environment. Different Blockchain platforms are discussed and further study can be done to apply the Blockchain in practical with the cloud platform. Ilya Sukhodolskiy and Sergey Zapechnikov [57] ensure privacy in cryptographic operations without the participation of cloud owners. It puts more emphasis on the access control mechanism of the cloud. Cloud access can be controlled using CP-ABE which is implemented on Ethereum. It ensures security by storing only hash-based cipher texts. The author Vorameth Reantongcome et. al. [58] has proposed in their paper about multi-tenancy co-resident attack which is caused due to leakage in data by a malicious tenant. The authors implicated a truffle framework to implement Blockchain with Ethereum. Ethereum encoded in solidity along with the smart contract is used. The transactions between the tenants and the cloud owner are recorded using the Blockchain. Further work can be done to improve the integrity and the confidentiality of the proposed model. Jin Ho Park and Jong Hyuk Park [59] find a solution for securing bitcoins by surveying Blockchain technologies. It installs an electronic wallet in the cloud for using the service using a secure bitcoin protocol and after using the service it successfully deletes the user details from the wallet. Public key encryption is used for encrypting the data. It verifies users' privacy by completely removing the wallet's details from the cloud. Researchers can study the future risks of using bitcoins. Ingo Weber et. al. [60] evaluate quantitative and qualitative analysis to

design a multitenant scalable architecture. The evaluation is based on integrity and isolation in the tenant's performance. Ethereum is used with Laava's industry partner for implementing a proof-of-concept prototype. Low cost, data integrity, and performance isolation can be achieved by the proposed architecture. Flexibility in anchored chains can be evaluated as a future scope. Meet Shah et. al. [61] used IPFS (InterPlanetary File System) protocol to emphasize data utilization, decentralized data storage, security, and privacy in their paper "Decentralized Cloud Storage Using Blockchain". It stores AES (Advance Encryption Standard) encrypted files on peer networks using IPFS protocol in the Blockchain guarded by smart contracts. Cryptocurrency is transferred to the peer's wallet from the user's wallet. The Blockchain's decentralized approach discussed in this paper is considered to be safe and secure. Wenlei Qu et. al. [62] proposed a technology for electronic voting in their paper which focuses on security and privacy issues by combining Blockchain and homomorphic signcryption. Homomorphic signcryption technique in Hyperledger Blockchain type is used to secure the voting ballots. Valid and invalid votes after aggregating the voting results are categorized using a smart contract. It improves and secures the voting process by reducing the time taken for the process in a secure and transparent manner without the need of any third party. Security can be further improved by practically implementing the proposed model and researching the model in detail. Sharath Yaji et. al. [63] proposed a technique using PHE schemes with Blockchain in their paper to preserve privacy. It focuses on attacks on the wallet, collision attack reimages attacks in the Blockchain. Goldwasser-Micali and Paillier PHE schemes are used in the proposed model which can bypass most of the attacks. The time required to process the model is comparatively less and is more secure. Latest attacks can be analyzed and experimented on in the proposed model with improved PHE schemes. Morampudi Mahesh Kumar et. al. [64] concluded a privacy algorithm to focus on malicious attacks, which is based on BMIAE (Blockchain multi-instance iris authentication using ElGamal Homomorphic Encryption). It is mainly designed for an untrusted server. Distance compute can be performed using a smart contract in BMIAE, where Elgamal is based on the hardness of these discrete problems. The proposed model guarantees confidentiality and integrity can be achieved along with the decreased execution time and the computational cost. Bobo Huang a et.al. [65] proposed BPS

which is the Blockchain-Publish-Subscribe model used to secure pub/sub streaming models from the attacks on the edge cloud caused by multiple tenants using the same pub/sub system. The illegal or unauthorized behavior by a tenant or an unauthorized user can be detected using the BPS model which is based on Blockchain functionality. It verifies the integrity using a Merkel tree and keeps a record of all the tenants in the access control list combined with the smart storage feature. It is analyzed that the proposed model outperforms security with a minimum overhead of performance. BPS can be tested to support complex deployments and improve scalability. Leila Ismail et. al. [66] analysed in their paper a new paradigm combining cloud-based services with Blockchain technology to provide an efficient and patient-centric view to the healthcare stakeholders. A BcC (Blockchain-cloud integration) is deployed for managing the patient details efficiently in the database management system for the healthcare industry. Further, this paper discusses the strengths and weaknesses of BcC architecture. BcC can overcome the individual shortcomings of cloud and Blockchain technologies; as the cloud provides a centralized service that may violate the privacy of the patient and Blockchain is not scalable and it faces some challenges with its efficiency. Future scope is to enhance the existing model for better scalability and efficiency. Gaopeng Xie et. al. [67] integrated Blockchain technology with the cloud data integrity verification scheme. It focuses on cloud data integrity and analyzed previous studies and works on improving the defects detected in the research and the user verification process is simplified. The authors introduced a lattice signature algorithm to resist quantum computing technology and combined it with a cuckoo filter for simplifying overhead in the computational process. It relies on a SIS (small integer solution) assumption to cope with the attacks. The proposed scheme can cope with quantum attacks and malicious attacks with high efficiency. To explore more features of the data integrity scheme combined with Blockchain technology. Caixia Yang et. al. [68] designed a Blockchain-based access control framework by integrating cloud computing. The access control permission of the data on the cloud is redefined, which is stored in the Blockchain. The proposed model overcomes the limitations of the Blockchain and the cloud. AuthPrivacyChain is the framework designed by the authors based on EOS (Electro-optical system) Blockchain. The proposed model is compared with the traditional cloud using the test tool JMeter. The users with proper access rights

can only access the data. The designed model can prevent the data from insider as well as outsider attacks. Wenjuan li et. al. [69] surveyed Blockchain-based trust models. Blockchain's decentralized approach helps in protecting the data from breaches as the traditional cloud trust model is not transparent and follows a centralized approach that cannot be trusted. In cloud computing, the efficiency and adaptiveness can be improved by using a hybrid edge cloud with double-Blockchain technology. Blockchain successfully builds trust by using a transparent approach, it avoids data leakage and eliminates the single point of failure because of its decentralized approach. In the future, Blockchain must be studied to collaborate with the new cloud technologies like edge computing, IoT applications, fog computing, and so on. The resource constraint may create a problem if all the data is stored on the chain, it may increase the processing time. Hence, it is to be researched. Mueen Uddin et. al. [70] reviewed the common cloud vulnerabilities that are mostly based on the virtualization platform, the identified issues are solved using the Blockchain-enabled models. The common vulnerabilities like a centralized security risk, transparency, resource sharing in a virtualized environment are discussed. The challenges with the Blockchain model are also discussed in this paper. Blockchain helps the cloud service providers in creating a virtual database and using a one-click method for accessing the services. Ketki R. Ingole and Sheetal Yamde [71] analyzed the business opportunities in both financial and non-financial sectors using Blockchain technology. The processing of bitcoin is discussed; attackers may steal the private key stored in the user's computer hard drive to hack bitcoin. Hence, the records stored in the computer must be deleted after use. The longer it stays in the user's system, the more is the data at risk. The authors of this paper suggested a way to secure the data from breach by simply removing the user's information from the system completely.

#### **CPABE technique:**

Hassan El Gafif and Ahmed Toumanari [72] proposed in their paper ciphertext-policy attribute-based encryption (CP-ABE) key encapsulation mechanism. **This scheme reduces the expense for the encryption operation as in traditional CP-ABE.** The authors proposed two CP-ABE mechanisms, one for the untrusted ABE service provider and another for the semi-trusted ABE service provider. The proposed schemes

are found to be secured under CPA and are considered as more efficient than the traditional CP-ABE scheme. PanJun Sun [73] evaluated some technologies related to cloud security like CP-ABE, KP-ABE (Key policy attribute-based encryption), proxy re-encryption, access control, and multi-tenancy and suggested to include the upgraded and updated technologies and policies to secure the data. The real-time execution of the analyzed methods is left as a future scope.

#### **Access control techniques based on role of the user:**

Rai [74] discussed different access methods which provides an efficient way to ensure only authorised users can access the data. Different access methods analysed are DAC (Discretionary Access Control), MAC (Mandatory Access Control), RBAC (Role based access control), TBAC (Task based access control) and ABAC (Attribute based access control) models. Traditionally, MAC and DAC were considered as trustworthy but RBAC method is used mostly by the organizations to protect the cloud data because of its advantages. The authors Kumar and Chatterjee [75] designed access rules for the E-health cloud based on the trust degree of the users to provide customized services to the users. These access rules are stored in the access rule database. Based on the user's category, the access view can be full, partial, or no view. A dynamic and adaptive access model is formed by adding the trust factor of the user in CPN tool. Proper verification of these access rules is required to check whether they follow the access control properties. Chhabra et. al. [76] compared the available access control techniques analyzed their suitability to the cloud computing environment. A multitenant environment requires fair access control between its tenants. ABAC technique is discussed in detail. ABAC grants access rights based on the policies that combine the attributes. The attributes are the basic building blocks and may contain static values like the role of the user. The future scope of ABAC is evaluated and suggestions for strengthening ABAC are analyzed. The authors Sethi et. al. [77] integrated an access control technique that is based on the role of the user. Each user requires a customized set of services; hence granting all the resources is not advisable. Administrator is responsible to assign the role of the user based on the trust factor. A spatio-temporal RBAC model grants the access on the basis of time of the request and the location of the user. This model is integrated with the Homomorphic encryption algorithm. The

proposed model evaluates the impact of the trust factor to strengthen the cloud security. The authors Ding and Yan [12] proposed a flexible way to control the access using a privacy-preserving data processing scheme. **Homomorphic encryption algorithm solely cannot secure the cloud data as it is a single key system, it cannot flexibly handle the data sharing and access control mechanism over the encrypted data.** The authors of this paper proposed an efficient and flexible way to control the data access in a privacy-preserving system. Paillier's partial homomorphic encryption is used with the cooperation of cloud service provider and the computation party. A total of seven operations are analyzed using Paillier's algorithm on outsourced encrypted data. These are Addition, Subtraction, Multiplication, Sign Acquisition, Absolute, Comparison, and Equality Test. The proposed attribute-based encryption scheme is tested and compared with existing work and it is found to be a better alternative. Future scope is to realize more operations, reducing the latency and improving the overall efficiency by applying edge computing and some pre-processing methods. The authors Hingwe and Bhanu [78] proposed a role-based access control that is integrated with PHE Paillier's algorithm for a multi-tenant database. A role hierarchy is maintained with access privileges and a session key is used for managing the active session. The session key is generated using SHA-2 (Secure hashing algorithm) to encrypt the data at client end and Paillier algorithm encrypts the sensitive data, hence data is protected during execution and in the database as well. Data integrity and confidentiality are achieved using the proposed model for access control as it protects data from SQL injection and privilege escalation. Role hierarchy management with least privilege grants used in the proposed method, does not allow privilege escalation and hence protects the data in an efficient manner. Though, user has to remember two or more keys which is considered as lagging issue with the proposed model and future work can be done to reduce this issue.

### **Using Multi-bits:**

Yu et. al. [79] proposed a **multi-bit public key encryption** protection based on LPN (Learning parity with noise). The proposed approach solves the encoding error to some extent. The scheme is compared with RSA and Damgard's multi-bit scheme, it is found that encryption time in the proposed scheme is slow than RSA while fast than

Damgard's and decryption time is faster than RSA while slower than Damgard's scheme. The proposed scheme can bypass quantum attacks effectively and decryption error problem is solved. **The scheme lags in achieving stronger CCA security.** In future, providing stronger CCA security is the main focus with smaller public key, ciphertext size and low computational overhead.

### **Hybrid approaches:**

The authors [80] proposed a combination of encryption algorithms AES, ECC and RSA to improve the security in the cloud. Two methodologies were used; first one proposed cryptographic algorithm and second one is the practical implementation of the algorithms for performance evaluation. It is found that AES takes less time in terms of encryption and decryption time while proposed hybrid algorithm takes more time in execution but secrecy is better here hence is more secure. Future scope is to find a more secure combination of algorithms with increased performance and least encryption and decryption time. Z. Saeed [32] proposed a cryptographic algorithm with 3 layers which is composed of AES, ECC and RSA methods. Java NetBeans is used to implement RSA-ECC algorithm and is compared with RSA-AES algorithm; it is observed that the time taken for encryption and decryption process by the RSA-ECC algorithm is comparatively less. Zaineldeen and Ate [81] proposed a hybrid approach to secure the cloud data. The homomorphic encryption schemes EHC and AES are integrated to form the hybrid technique. It is found that the proposed methodology is better in terms of encryption-decryption time, memory requirement, throughput measurement and power consumption used by the techniques. Kardi and Zagrouba [82] proposed a hybrid approach using RSA and ECC algorithm schemes for a wireless sensor networks (WSNs). The hybrid approach overcomes the weakness of both the algorithms, resulting in a compatible and more efficient scheme. Some findings in the proposed study states that asymmetric cryptosystems server better WSN as it offers better security than symmetric cryptosystem which take less memory and is faster comparatively. Hence, asymmetric cryptosystem combining ECC and RSA is developed which is more flexible and configurable to user's customized needs based on different nature of the applications used. Future work is to explore the proposed algorithm for different applications using various technical configurations. The authors



Abid et. al. [83] proposes an optimized HE CRT-RSA algorithm to overcome the challenge with traditional homomorphic encryption algorithm of fast decryption and slow transmission. Multiple keys are utilized for better communication and security. The proposed model is compared with classical RSA algorithm and it is found that the proposed model 3 to 4% faster than the RSA algorithm and is more secure.

## 2.2 Comparison of past proposed and proposed model complexities based on number of computations

K. Kumari et. al. [84] proposed two schemes based on homomorphic encryption algorithm, the first one is based on Carmichael's Theorem and the second one is an improved version of EHC algorithm. Both the schemes, CTHE (Carmichael's Theorem-based Homomorphic Encryption), and the MEHE (Modified and improved Gorti's Enhanced Homomorphic Encryption) reduces the noise using the modulus switching technique and are secure under quadratic residuosity, integer factorization and discrete logarithm problems. The complexities as evaluated by the authors are presented in *Table 2.1* [84].

Table 2.1 Computational complexity of various Homomorphic algorithms

Schemes	CTHE	MEHE	Gorti's EHC scheme	DGHV scheme	Paillier	ElGamal
Number of computations	$7E_k+1D_k$	$6E_k+1D_k$	$5E_k+1D_k$	$5E_k+2D_k$	$5E_k+6D_k$	$6E_k+3D_k$
Overall time taken in encryption, evaluation and decryption process (ns)	$5.3 * 10^{11}$ for $n= 2^{11}$	$7.9 * 10^9$ for $n= 2^{11}$	$6.6 * 10^8$ for $n= 2^{11}$	$7.7 * 10^9$ for $n= 2^{11}$	$8.3 * 10^{13}$ for $n= 2^{11}$	$5.6 * 10^{12}$ for $n= 2^{11}$
$P_uK(n/n^2)$ length	$2^{11}$ to $2^{18}$	$2^{11}$ to $2^{18}$	$2^{11}$ to $2^{18}$	$2^{11}$ to $2^{18}$	$2^{11}$ to $2^{18}$	$2^{11}$ to $2^{18}$

As shown in *Table 2.1*, the comparison of the proposed algorithms (existing and current) is shown based on the number of computations, time taken and the public key

length. It can be observed that Gorti's EHC and DGHV schemes computation are close to MEHE and CTHE schemes. Since Gorti's scheme lacks in noise reduction, the proposed schemes MEHE and CTHE are considered. Though, some hybrid approach is still required since the schemes MEHE and CTHE are basic structure state of the art schemes and the implementation is not available in the libraries and packages. Hence, some hybrid approach is required which can work on a larger real number space without increasing the overall computational cost. Blockchain technology can be embedded with the proposed model to ensure mutability.

In the proposed model, Gorti's EHC algorithm is considered and is integrated with BGV algorithm with key mutation and mapping techniques. Any homomorphic scheme is said to be suitable for real-world applications if it reduces computational and storage overhead. EHC takes very less storage space and the computational complexity is better as discussed in *Table 2.1*. The total cost of the proposed hybrid EHC-BGV model is comparatively less and the proposed model supports both static and dynamic data. Though, there is still scope of improvement.

### **2.3 Issues with current methodologies and Research gaps**

Most of the researchers focused on the data storage correctness (using Homomorphic token), but a very few talks about data transmission while it is in transit-mode, buffer-storage-mode, end-user mode. Some rely on generating a random token number every time a user log in, in addition to username and password[11][42]. The existing approaches and models used token-based models along with dual authentication methods [34][38][43][37]. Some of the researchers integrated CP-ABE[85][72], proxy re-encryption [4][73], KP-ABE [73], JSON based model with HTTPS/TLS transmission [10], SLA oriented multi-tenant hybrid scheme [20], FHE schemes such as AHEE scheme [15], NEHE [8][14] and so on. While others focussed upon validation along with verification-based approaches for accessing the data and control between the tenants [52][67].

Some authors used FHE schemes with tenant isolation [1][25]. Homomorphic encryption algorithm is considered as a safer option for encrypting the data by

maintaining the privacy, but it alone cannot provide data privacy and control data access both at the same time [53][86].

Some hybrid techniques are also developed as discussed in the related work but none of the schemes results in a more secure way of protecting the data without any weaknesses or lagging issues [42][82][81].

Some of the issues in the present schemes includes CCA security issue, computational overhead, limited to single-bit or using multi-bit alone is also not sufficient [24][78].

Some researchers used a multifactor authentication method for verification and validation of the user to protect them from unauthorized access. Though it can secure the cloud data from the traditional threats, but for upcoming threats, this method is not considered as safe [42]. A multifactor authentication method may combine the verification factors like login credentials, OTP, and salting. Salting is dependent on the operating system used and hence cannot be considered as safe [42].

The attacker can guess the password using brute force attack or any other related attacks [21][42]. Hence, there must a mechanism for shuffling the password based on some parity rules and not dependent on the operating system like salting.

A key mutation concept is currently in use by security experts. It shuffles the keys automatically, hence it becomes difficult to access the data where password changes automatically using some parity rules [87][40][41].

Existing role-based access control methodologies and technologies are not sufficient to secure a multi-tenant cloud environment [74][75][77][12][78]. Hence, a new technique is required to protect the cloud data.

## **2.4 Addressing the issues**

We proposed a role-based authentication multi-factor model with a customized tenant environment using FHE algorithm and CPABE technique. A combination of FHE algorithms EHC and BGV is used for token and keys generation with parity mapping and key mutation concept introduced.

The proposed model secures the cloud data from CCA attacks as EHC is IND-CCA. It protects the data from the insider and the outsider attacks both, by classifying the users based on their roles using CPABE algorithm. The multiple tenants are categorized into different layers based on their role using a CPABE model which is an inexpensive method [72] and EHC requires less memory and power [3][16] which results in less investment on storage resources. Hence, it can be concluded that the proposed research model is inexpensive. In terms of computational time, BGV takes less time than most of the past proposed algorithms like RSA and AHEE [3] and the processing time in EHC is comparatively less [17]. The key mutation technique is embedded with the hybrid model to provide better security when attacker tries to access the data.

## **2.5 Brief summary of this chapter**

This chapter discusses the issues related to the past proposed methodologies and presents a solution to address the problems identified.

## Chapter-3

### Methodology of the proposed research work

#### 3.1 Objectives of the proposed work

The research aims at improving security and privacy against the most vulnerable threats of cloud computing. Main objectives of the research can be defined as follows:

- To analyse existing Homomorphic techniques in multi-tenant environment.
- To design multi-tenant access control logs in Homomorphic encryption technique.
- To develop Homomorphic encryption technique in multi-tenant cloud environment.
- To test and validate the developed technique.

#### 3.2 Implementation of Ciphertext Policy Attribute-Based Encryption (CP-ABE)

##### Algorithm

CP-ABE algorithm consists of four algorithms namely, Setup, Encrypt, KeyGen, and Decrypt. [88][85]

**Setup:** The public key and master key are generated using the bilinear group with a prime number and two random generators. The input is based on security parameters and the universe of the attribute. It takes no other input than the implicit parameters.

**Encrypt (Public\_key\_PK, Message\_M, Access\_tree\_T):** Input values are public key, message, and a tree-like access structure. The message can be encrypted using an access structure like a tree. A polynomial is chosen for each node in a top-down manner in such a way that, degree of each node is one less than the threshold value. The ciphertext can only be accessed if the user satisfies the attribute set for the access tree.

**KeyGen (Master\_key, Attribute\_set\_S):** It generates the private key by taking as input the master key and attribute set S, describing the key.

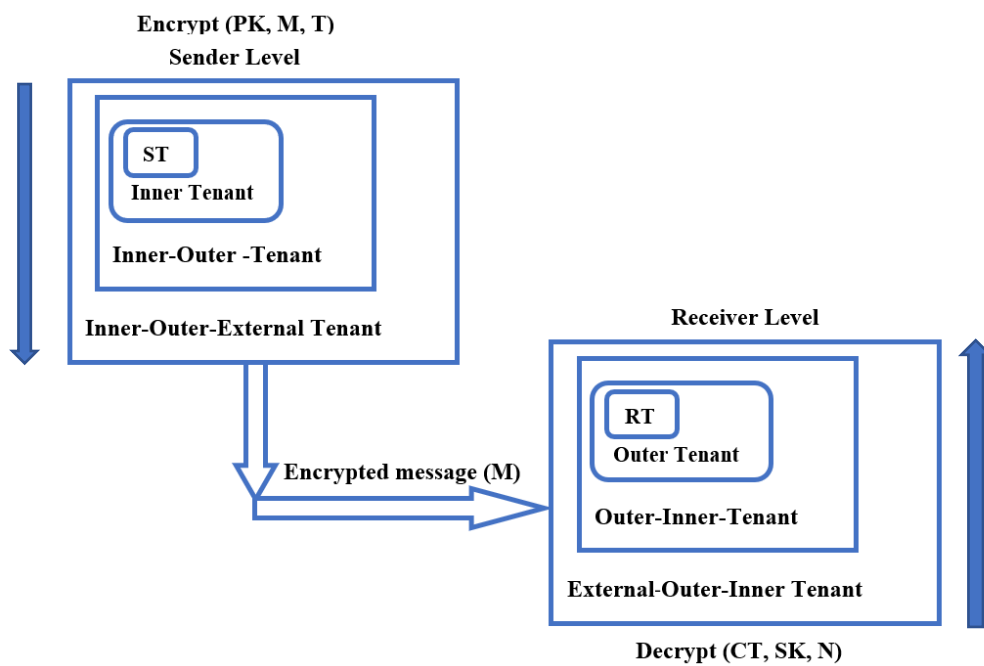
**Decrypt (Cipher\_text\_CT, private\_key\_SK, node\_N):** Original message can be decrypted using a combination of public and private keys, access tree structure, and

ciphertext. Based on the condition of satisfying the set of attributes S by access tree, the ciphertext can be decrypted.

Each node in the access tree can be evaluated using the expression:

$$\text{node} = \text{hash} \sum (\text{node.children}(N).\text{hash}) \quad \dots(1)$$

Here, a node is integrated with the help of encrypting the hash table by incorporating the local node. Hash is a key that can be either a token key or salting or a public key. Here, a node is a communicating device and children are tenants which belong to a user or an organization. It authenticates data using hashing technique in an MTE.



*Figure. 3.1 Ciphertext Policy Attribute-based Encryption (CP-ABE) algorithm in multiple tenants*

In *Figure 3.1*, multi-tenant architecture is created where Sender Tenant (ST) sends a message to Receiver Tenant (RT) by encrypting it using the CP-ABE algorithm[89]. The ST transfers the data to the Inner Tenant, from where the data is sent to the Inner-Outer-Tenant and finally to the outer layer of sender end that is, Inner-Outer-External Tenant. The data or packet is then sent to the network which is received by the receiver's outer most tenant layer that is, External-Outer-Inner-Tenant level and from this level the data is transferred to the Outer-Inner-Tenant which transfers it to the Outer Tenant, which is then received by the RT. Homomorphic encryption can be applied for better

security. A homomorphic encryption algorithm can be used in combination with the CP-ABE algorithm to make it more secured since CP-ABE model divides the tenants to multiple levels, providing a customized access approach for each tenant. It allows any number of operations on the ciphertext itself.

For two enterprise communication, according to the relation of source and destination (tenants) levels, the possible and relevant multiple tenants between two enterprises are created as an inner tenant, outer tenant, Inner-Outer -Tenant, Inner-Outer-External Tenant, Outer-Inner-Tenant, External-Outer-Inner Tenant. Although, these tenant levels can be extended for more than two enterprises.

Sender tenant: The sender tenant is the user or organization that wants to send a packet(s) to another tenant at the enterprise level. A sender tenant uses a token along with the public key for encryption. It acts on the Internal-Global setup.

Inner tenant: It is a tenant which lies in the sender tenant block. Internal-external setup is required for this level. It is completely based on token only. No keys are required for internal communication among tenants.

Inner-outer tenant: It is based on an Internal-External-Global setup. It uses token, salting and public key for encrypting the message. For key generation access tree (collection of nodes or tenants) is used with message and ciphertext along with the token, salting and public key. The decryption of ciphertext can be done using a secret key with salting and token.

Inner-Outer-External Tenant: It is based on the Internal-External-Global-Global setup. Here, the packets are transmitted outside the enterprise. For encryption, token, salting, public key, message and access tree are used. A combination of public and private keys is generated in this setup. Decryption is done based on nodes, and token, salting and a private key are used on the ciphertext for decryption.

External-Outer-Inner Tenant: It is based on a Global-Global-External-Internal tenant setup. It is responsible to receive the packets from the sender enterprise tenant. Encryption of message or packet on access tree is done using salting, a public key and a token key. A combination of keys is generated along with public and private keys

based on nodes. Decryption is based on nodes and can be done using a private key, token and salting on the ciphertext.

Outer-Inner-Tenant: It is based on the Global-External-Internal setup. It takes the packets from the outermost layer of the tenants and sends them to the outer tenant layer which is next to the receiving tenant. A public key, salting and token key are used for the encryption process and for decryption, a private key, salting, and token key are used.

Outer tenant: In this level, a packet can be received and decrypted using the salting technique, which adds some random collection of digits, symbols, and alphabets in the password field to make it difficult to guess by a hacker or an intruder. It used salting, secret key, or private key and token to decrypt the ciphertext received. It acts on an External-Global setup.

Receiver tenant: The tenant for whom the packet is meant can receive the packet and ciphertext can be decrypted using a combination of private key and token on the nodes.

Table 3.1 CP-ABE- Multi-tenancy parameters

<b>Multitenant / CP-ABE</b>	<b>Setup</b>	<b>Parameters</b>	<b>Encrypt</b>	<b>KeyGen</b>	<b>Decrypt</b>
Sender Tenant	Internal-Global	Token, PK, M, T	Token, PK, M, T	PK, M, T	CT, SK, N, Token
Inner - Tenant	Internal-External	Token-based	Token-based	Token based	Token based
Outer Tenant	External-Global	Token + Salting based	Token + Salting based	Token + Salting + CT based	Salting + Token + CT based
Inner-Outer - Tenant	Internal-External-Global	Token + Salting + PK based	Token + Salting + PK based	Token + Salting + PK + M + T + CT based	Token + Salting + CT + SK based
Inner-Outer-External Tenant	Internal-External-Global-Global	Token + Salting + PK + M + T based	Token + Salting + PK + M + T based	Token + Salting + PK + M + T + CT + SK + N based	Token + Salting + CT + SK + N based
External-Outer-Inner Tenant	Global-Global-External-Internal	T + M + PK + Salting + Token	T + M + PK + Salting + Token	N + SK + CT + T + M + PK + Salting + Token based	N + SK + CT + Salting + Token Based
Outer-Inner-Tenant	Global-External-Internal -	PK + Salting + Token based	PK + Salting + Token based	CT + T + M + PK + Salting + Token	SK + CT + Salting + Token based
Outer Tenant	Global-External	Salting + Token based	Salting + Token based	CT + Salting + Token based	CT + Token + Salting based



Inner - Tenant	External- Internal	Token-based	Token-based	Token-based	Token based
Receiver Tenant	Global- Internal	PK, M, T, Token	CT, SK, N	PK, M, T	Token, CT, SK, N,

The *Table 3.1* describes the parameter generation at various levels in a CP-ABE algorithm. The terms used in the table are T- Access tree, PK – Public key, SK – Secret key or private key, M – Message, CT – ciphertext, N – Node in a tree,

For Internal-Global and Global-Internal tenant setup, token key ‘Token’ is used. Since for internal communication, only token keys can be used with no need of using an extra combination of keys which is required for global communication. An additional key with the token key is sent along with the packets in such communications.

For encryption, token, salting, and public keys are used in sender enterprise to encrypt the packets. In key generation, a secure key is added to send packets to another enterprise tenant and a combination of parameters like token, salting, public key, private key or secret key, ciphertext, and N-based are used.

For Internal-External and External-Internal tenant setup, communication is done locally so an additional key is not required here. Token key is sufficient to provide security.

For all other tenant setups, salting is used along with the token keys to provide an extra layer of protection that is multi-factor authentication mechanism is used here.

For example, some financial institutions are already using such secured systems which require multiple levels of authentication from the user. It requires a combination of username, password, OTP, salting technique combined with encryption, or an extra level of authentication added for better security.

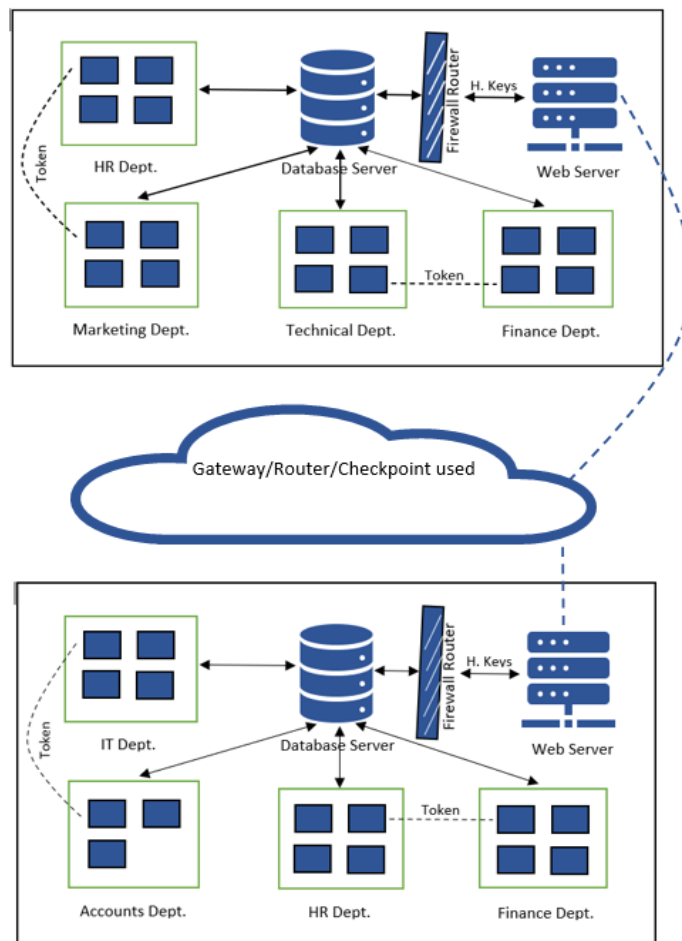
The additional set of keys result in better security when communicating at a global level tenant mechanism.

### **3.3 Designing the multi-tenant multiple-enterprise model**

In the proposed architecture, the internal communication between the tenants is done by the main server or database server. In a single enterprise, tokens are used for internal verification as multiple tenant’s works on the same LAN. The communication outside

the enterprise is done using a web server. Storing data on cloud using tokens only is not considered as a secure way. FHE algorithm generates an additional key which is disabled by default and will be active only when an attacker tries to intercept or access the data. Communication between different enterprises requires a firewall router to ensure the data privacy and security.

The communication server used is web server which means the IP address is visible to the attackers, and hence extra security protocols are required, while the main server is working in the background where IP address is not visible. The IP address of which is not known to intruders. *Figure 3.2*, represent the enterprise-level architecture in a multi-tenant environment.



**Figure 3.2 Enterprise-level multi-tenancy environment for data access control, authorization, and authentication**

In *Figure 3.2*, one main database server is required for each enterprise. Therefore, a total of 2 main servers and 2 communicating servers are needed for sharing data. There are a total of 4 departments in each enterprise, hence, total 8 departments with 16 users in the first enterprise while 15 in the second one. A total of 2 firewall routers, which can be Cisco routers or simple checkpoint devices, can be attached between the main server and the web server. Its main focus is to protect and secure data from unauthorized access. The multi-tenant cloud architecture communicates in two different ways; one for the insider tenant and another for the outsider tenant. Further these insider and outsider tenant are categorized into different sub-tenant layers as explained in Section 3.2 in the CP-ABE model. For insider tenant, token is generated while for outsider tenant, key is generated; this key is generated based on the outsider user role or type. For trusted outsider tenant category, private key is generated; for untrusted tenants, type of user or tenant is further categorized into partial tenant and guest tenant, hybrid key is generated for a partial tenant and public key is generated for anonymous or guest tenant. Hence, multiple-tenant layers are designed on the cloud interface and data packets can be transmitted securely using the proposed hybrid methodology integrated with self-key mutation technique.

### **3.4 FHE Blended Schemes EHC and BGV based Environment:**

The FHE scheme ensures that the data computations are evaluated on the encrypted state of data with no requirement of decryption operation [16]. Hence, it is considered to provide better security to the cloud data and services.

The proposed methodology uses EHC and BGV algorithm schemes for generation of keys.

In the proposed model, EHC FHE scheme is used for communication inside the organization and generates tokens for this purpose. As communication within the enterprise is considered safe; generating only token is sufficient for data transmission.

The token will expire after some specific period of time depending on the type and role of the user and may have different access permissions for a file [78]. For example, an HR head may have different rights than a technical head for the same file. Hence, token

life time is used to provide more security in the organization. Another example is with banking institution, where the login session may expire after some time period due to the inactivity or idle state of the user. In a similar way, the token expires after a specified time period based on the user role type.

For an outsider user, BGV algorithm is used for partial user only, and for other cases, EHC algorithm is used. A hybrid key is generated for a partial user using BGV algorithm. This key is a combination of keys. Key expiration takes place for partial user category as this user role is not considered safe.

An additional key is generated using hybrid EHC and BGV algorithm in addition, that is hidden by default and will be active for the attacking scenario only. Suppose one PK and two PuK's are generated using FHE algorithm then a hidden key with size 4-bit is generated using bit mapping technique. This hidden key is sent along with the other public and private keys. This key will only be active in case of packets being attacked by the attacker.

This hybrid approach gives different layers of security that can protect the data in an efficient manner. For a distributed enterprise, a hybrid approach is more suitable since different categories of users are there, based on their roles, the data access services are provided.

### **3.5 Automated Key Filter and Bit-Mapping Techniques using Hybrid EHC-BGV Homomorphic Algorithm**

The proposed research work is extended from the traditional BGV and EHC algorithms. The traditional cipher substitution techniques are extended and integrated with the self-key mutation technique. The proposed hybrid BGV-EHC model is synchronized with the user role to generate the tokens and keys for securing the data on the cloud. BGV is used at places where static and dynamic data both are used such as if the role of the user is partial then BGV algorithm is used to generate hybrid key; a partial user may be working on static data but for some cases the user may need access to the dynamic data. For example, a user may want to access data of the same organization but in different branch location then the dynamic allocation is assigned to the user. EHC takes low

memory space, hence it is used at places where memory consumption is high. Since 56 parity rules are generated, therefore some complexity may increase if rules are further created. Still to handle 56 parity rules, some storage is required as resources are required to operate the required computations on these rules. Hence, EHC and BGV both are used in this case to handle static and dynamic data with less storage. EHC is used as major algorithm and BGV is used as minor algorithm. In the proposed model, the sender and receiver's properties are shared for the transmission of data on the cloud. The parameters shared are sender & receiver's name and ID, the type of data being shared, tenant type, role type, authentication type, data size bytes, service type, method type, session initiation time and session end time.

The selection of the algorithm used for keys and token generation depends on the type of the user or tenant [77][78]. A tenant can be within the organization or outside it. An insider tenant requires only a token for the secure transmission of the data while an outsider tenant requires additional security for data protection.

*Figure 3.3* describes the hybrid approach using BGV and EHC algorithms. Depending on the parameters initialized, the user's category is identified such as Sender's name, Sender's IP address, Receiver's name and IP address, type of tenant etc. Moreover, admin have the power to grant roles to the tenants. The role of the user is divided into two categories; insider and outsider category and based on these categories the respective process for implementation follows.

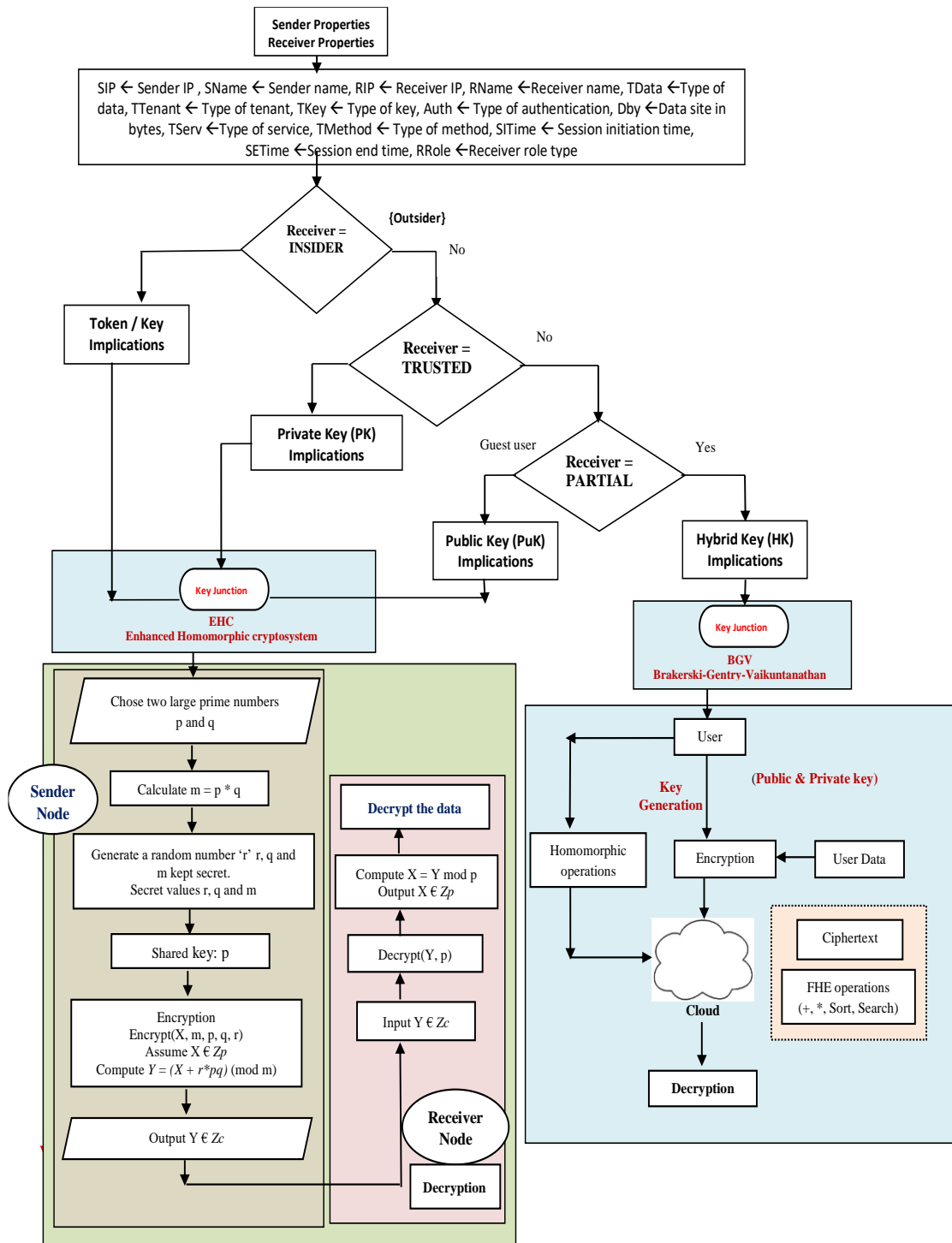


Figure 3.3 Hybrid approach using BGV and EHC Homomorphic algorithms

If the receiver type is insider, token/key implications are executed using the EHC algorithm. The insider tenant is considered as the most secured cloud environment in the organization. The token is generated randomly and expires after some period of time

to provide a secure architecture. Similar way, if the receiver is trusted but comes under outsider category, private key implications follow and if the receiver is non-trusted and comes under the partial category of the user, a HK is generated using BGV key junction. A hybrid key is a combination of PuK and PK based on some parity rules. The last type of tenant is for the anonymous category, public key is generated in this case using EHC key junction. Hence, the key junction for generating the public, private keys and token generation is EHC and the key junction for the hybrid key is BGV.

Further the hybrid model is integrated with the self-key mutation technique which will be active only in case of an attack and different parity rules are designed and applied for bit-mapping during the mutation process. The mutation technique is used for shuffling the password keys such that the attacker cannot guess the password. If anyhow, the password is known to the attacker then the system checks the registered IP address of the device used. This IP address is not found in the dictionary where authorized user's data is saved, hence it blocks the communication and shuffles the key again using the mutation technique. Any new device must be registered first, before using it to access the data on the cloud.

Hence, the proposed research model automates the key filtering and bit mapping techniques based on the user's or tenant's role.

### **3.6 Secure Token and Key Implications based on Dependable and Non-Dependable Factors**

The complete process of communication relies on the type of the user category [90], the category can be insider or an outsider. Figure 3.4 explains the generation of token and the keys depending on the role of the user. The role or category of the user is identified by checking the parameters values initialized at the beginning of the flowchart. It includes the details like Sender's name IP address or Receiver's name and IP address, type of the tenant, keys, authentication mode, or service and data site in bytes size, the receiver's role, session timings etc.

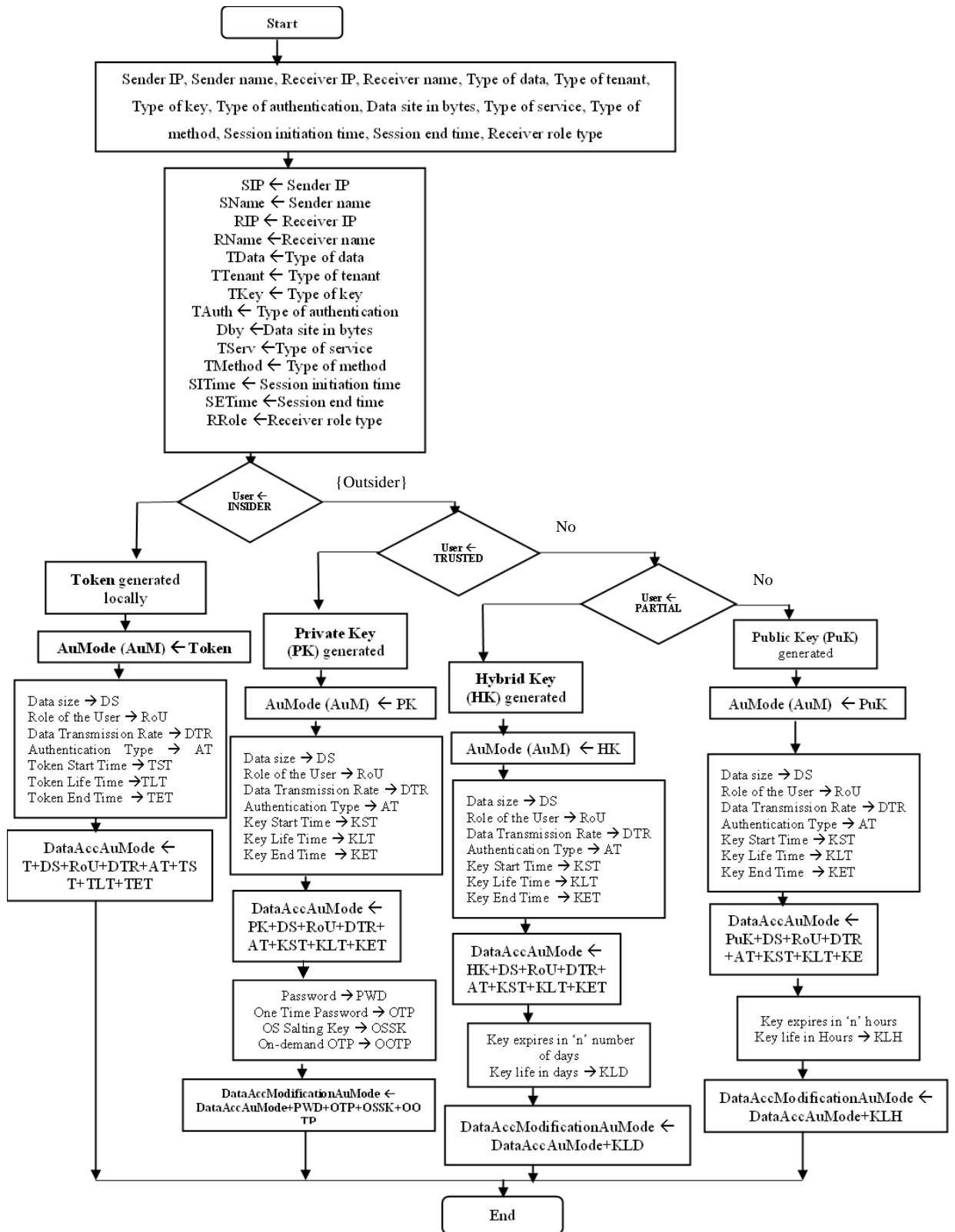


Figure 3.4 Steps for generating the token and the associated keys depending on the role of the user



For an insider user category, only token is generated. Token is sent in the authentication and the required parameters are sent for the communication process. Next, evaluation of the data access authentication mode takes place. The parameters for insider user are not modified since this user category has most of the access permissions.

The second case is when the user is an outsider. Additional security protocols are considered for this user category. It is subcategorized into a trusted user, partial-trusted user and anonymous or guest user.

A trusted user category requires additional checks for authentication and verification of the user. Different techniques are used for the verification process along with OTP generation. Parameters required for authentication process to access the data are shared. For the login process, along with the username, password, and Salting method, OTP is generated which is sent on the user's mobile number. Apart from using these authentication techniques, an additional key OOTP is generated, that is, on-demand OTP is generated that remain in deactivated state by default and will be active when an attacker tries to access the data packets, this key is generated using the bit mapping technique based on some production rules that are designed specifically for the attacking case scenario. The attacker is identified by checking the details like IP address, and location of the user from the dictionary. An invalid three login attempts, sends an email to the data owner with details of the attacker like IP address, location, and the file name being accessed. It automatically changes the password using self-mutation technique.

An outsider untrusted category is sub-divided into a partial user category where key expiration concept is added. The session of the user will remain active for some days and the token will expire afterwards. Firstly, a condition to check the role of the user is applied, if the user is a partial user, then a hybrid key is generated. The authentication mode requires a hybrid key and the data access authentication mode uses a hybrid key and the parameters required for token/key expiration. For a partial user category, the key will expire after 'n' number of days automatically. In data access modification authentication mode, the parameter changes take place. It updates the parameters in the database after processing them. It is a combination of data access authentication mode parameters and hybrid key and session time of key in days.

Finally, the last category of user is an outsider un-trusted user, which is anonymous user. This user role category is considered as least secure. The session will be active for an 'n' number of hours. Data can be accessed using a public key which is the least secure option. The data access authentication is done using the public key and the parameters passed in the previous step. And the parameters are updated in the next step.

### 3.7 Algorithm for generating the token/keys based on the user-role-type

The algorithm in Table 3.2 [91] defines the procedure for generating the keys by following the proposed methodology.

Table 3.2 Algorithm for generating the token/keys

<b>Step</b>	<b>Preparing the data for communication.</b>
<b>0:</b>	
<b>Step 1:</b>	Initiating the parameters for sending along with the data as, Sender IP, Sender name, Receiver IP, Receiver name, Type of data, Type of tenant, Type of key, Type of authentication, Data site in bytes, Type of service, Type of method, Session initiation time, Session end time, Receiver role type.
<b>Step 2:</b>	The dependable and non-dependable parameters are tuned with initiated variables as, $SIP \leftarrow sip\{s_1, s_2, s_3, \dots, s_n\}$ $Sname \leftarrow \text{Sender name}$ $RIP \leftarrow rip\{r_1, r_2, r_3, \dots, r_n\}$ $Rname \leftarrow \text{Receiver name}$ $Tdata \leftarrow tdata\{td_1, td_2, td_3, \dots, td_n\}$ $Ttenant \leftarrow \text{Type of tenant}$ $Tkey \leftarrow tkey\{tk_1, tk_2, tk_3, \dots, tk_n\}$ $Tauth \leftarrow tauth\{\tau_1, \tau_2, \tau_3, \dots, \tau_n\}$ $Dby \leftarrow \text{Data site in bytes}$ $Tserv \leftarrow tserv\{ts_1, ts_2, ts_3, \dots, ts_n\}$ $Tmethod \leftarrow tmethod\{tm_1, tm_2, tm_3, \dots, tm_n\}$ $SITime \leftarrow sitime\{sit_1, sit_2, sit_3, \dots, sit_n\}$ $SETime \leftarrow setime\{set_1, set_2, set_3, \dots, set_n\}$ $RRole \leftarrow rrole\{rr_1, rr_2, rr_3, \dots, rr_n\}$
<b>Step 3:</b>	User Integration: Categorization of the user, such that
<b>Step 3.1:</b>	If (user == INSIDER) (INSIDER $\leftarrow$ insider $\{i_1, i_2, i_3, \dots, i_n\}$ Token (T) generated (T $\leftarrow$ t $\{t_1, t_2, t_3, \dots, t_n\}$ )
	Parameters tuning to the variables,

---

$DS \leftarrow ds\{ds_1, ds_2, \dots ds_n\}$   
 $RoU \leftarrow rou\{rou_1, rou_2, \dots rou_n\}$   
 Data Transmission Rate  $\rightarrow$  DTR  
 $AT \leftarrow at\{at_1, at_2, \dots at_n\}$   
 $TST \leftarrow tst\{tst_1, tst_2, \dots tst_n\}$   
 $TLT \leftarrow tlt\{tlt_1, tlt_2, \dots tlt_n\}$   
 $TET \leftarrow tet\{tet_1, tet_2, \dots tet_n\}$   
 $DataAccAuMode \leftarrow T+DS+RoU+DTR+AT+TST+TLT+TET$

---

**Step** If (user == TRUSTED) (TRUSTED  $\leftarrow$  trusted {tr<sub>1</sub>, tr<sub>2</sub>, tr<sub>3</sub>, .. tr<sub>n</sub>})  
**3.2:** PK (Private key) generated (PK  $\leftarrow$  pk{pk<sub>1</sub>, pk<sub>2</sub>, pk<sub>3</sub>, .. pk<sub>n</sub>})

Parameters tuning to the variables,  
 $DS \leftarrow ds\{ds_1, ds_2, \dots ds_n\}$   
 $RoU \leftarrow rou\{rou_1, rou_2, \dots rou_n\}$   
 Data Transmission Rate  $\rightarrow$  DTR  
 $AT \leftarrow at\{at_1, at_2, \dots at_n\}$   
 $KST \leftarrow kst\{kst_1, kst_2, \dots kst_n\}$   
 $KLT \leftarrow klt\{klt_1, klt_2, \dots klt_n\}$   
 $KET \leftarrow ket\{ket_1, ket_2, \dots ket_n\}$

Accessing data,  
 $DataAccAuMode \leftarrow PK+DS+RoU+DTR+AT+KST+KLT+KET$

Login keys,  
 Password  $\rightarrow$  PWD  
 One Time Password  $\rightarrow$  OTP  
 OS Salting Key  $\rightarrow$  OSSK  
 On-demand OTP  $\rightarrow$  OOTP {active only in attacking scenario}

Parameters modified,  
 $DataAccModificationAuMode \leftarrow DataAccAuMode+PWD+OTP+OSSK+OOTP$

---

**Step** If (user == PARTIAL) (PARTIAL  $\leftarrow$  partial {par<sub>1</sub>, par<sub>2</sub>, par<sub>3</sub>, .. par<sub>n</sub>})  
**3.3:** HK (Hybrid key) generated (HK  $\leftarrow$  hk{hk<sub>1</sub>, hk<sub>2</sub>, hk<sub>3</sub>, .. hk<sub>n</sub>})

Tuning the parameters to variables,  
 $DS \leftarrow ds\{ds_1, ds_2, \dots ds_n\}$   
 $RoU \leftarrow rou\{rou_1, rou_2, \dots rou_n\}$   
 Data Transmission Rate  $\rightarrow$  DTR  
 $AT \leftarrow at\{at_1, at_2, \dots at_n\}$   
 $KST \leftarrow kst\{kst_1, kst_2, \dots kst_n\}$   
 $KLT \leftarrow klt\{klt_1, klt_2, \dots klt_n\}$   
 $KET \leftarrow ket\{ket_1, ket_2, \dots ket_n\}$

Accessing data,  
 $DataAccAuMode \leftarrow HK+DS+RoU+DTR+AT+KST+KLT+KET$

---

---

Key expiration time,  
if (KLT >= KLD) {for KLD as key life time in days}  
Key expires

Parameters modified,  
DataAccModificationAuMode  $\leftarrow$  DataAccAuMode + KLD

---

**Step** If (user == ANONYMOUS) (ANONYMOUS  $\leftarrow$  an {an<sub>1</sub>, an<sub>2</sub>, an<sub>3</sub>, .. an<sub>n</sub>})  
**3.4:** PuK (Public key) generated

Parameters tuning to the variables,  
DS  $\leftarrow$  ds{ds<sub>1</sub>, ds<sub>2</sub>, ..... ds<sub>n</sub>}  
RoU  $\leftarrow$  rou{rou<sub>1</sub>, rou<sub>2</sub>, ..... rou<sub>n</sub>}  
Data Transmission Rate  $\rightarrow$  DTR  
AT  $\leftarrow$  at{at<sub>1</sub>, at<sub>2</sub>, ..... at<sub>n</sub>}  
KST  $\leftarrow$  kst{kst<sub>1</sub>, kst<sub>2</sub>, ..... kst<sub>n</sub>}  
KLT  $\leftarrow$  klt{klt<sub>1</sub>, klt<sub>2</sub>, ..... klt<sub>n</sub>}  
KET  $\leftarrow$  ket{ket<sub>1</sub>, ket<sub>2</sub>, ..... ket<sub>n</sub>}

Accessing data,  
DataAccAuMode  $\leftarrow$  PuK+DS+RoU+DTR+AT+KST+KLT+KET

Key expiration time,  
if (KLT >= KLH) {for KLH as key life in hours}  
Key expires

Parameters modified,  
DataAccModificationAuMode  $\leftarrow$  DataAccAuMode + KLH

---

**Step** End if

**4:**

---

**Step** End

**5:**

---

### 3.8 Experimental Scenario and Analysis:

#### 3.8.1 Attacking scenario

The users or tenants are categorized into two groups; authorized tenants and unauthorized tenants. An authorized tenant has authenticated login credentials while an unauthorized user or an attacker does not have valid credentials to access the cloud. To identify the type of user, a dictionary is designed with details of all the authorized

tenants. The user must register his/her machine accessing the data just like the banking systems follow. Any new device must first be registered to use. The details included for registration are user's name, IP address, and location. The key shared after registration is in active mode but not enabled. The user must change the key from its default value to enable it.

An attacker is identified by using the dictionary containing details of all the authorized users. The entry of the IP address of the attacker is not found in the dictionary; hence it is recognized as an unauthorized user. The access to unauthorized users is blocked automatically using mutation technique.

Consider a scenario where an ex-employee wants to access the previous company's cloud data, his/her details may not be deleted from the dictionary. That means, the credential details may still be active, in this scenario the location of the user is verified and since the user's IP address is different, the access will be blocked.

### **3.8.2 Self-key mutation**

In case, an attack is identified using the dictionary technique, the self-key mutation takes place automatically. This is a process of shuffling bits is based on some rules [87].

Consider a scenario where an attacker can guess the password, considering 30 seconds are needed to guess a password or a character. If the attacker succeeds to guess a character, then in these 30 seconds, the self-key mutation process occurs. This process of self-key mutation repeats itself after every 30 seconds. Hence after every 30 seconds, the password will change and the attacker cannot access the cloud data leaving the cloud protected and secured. The authorized user can request for a new password again to access the file which the attacker failed to access. As, the user will receive an email after 3 invalid attempts by the attacker. The email contains the geographical information, IP address and other relevant details of the attacker.

The self-key mutation concept is used for randomly changing the position of the key bits to increase the complexity. Cloud security can be based on the proposed model for better protection from attackers.

### 3.8.3 Working of self-key mutation

The characters entered by the user are first converted to binary bits because the homomorphic encryption algorithm works on the binary bits only [41]. For conversion to binary bits, ASCII code is used. The characters are converted to ASCII code first and then decimal to binary conversion takes place.

Table 3.2(a) Primary data of generated key, salting values, general OTP for data access and data control, On-demand OTP for verification and validation, and instance of Data

Key	Salting	OTP		On-demand OTP		Data
		Data access	Data control	Verification	Validation	
8 bytes	4 bytes	4 bytes	6 bytes	4 bytes	6 bytes	64-bit of data in single instance

Table 3.2(b) Primary data of generated key, salting values, general OTP for data access and data control, On-demand OTP for verification and validation, and instance of Data

Key	Salting	OTP		On-demand OTP		Data
Pooja@25	Xe92	1723	201723	1723	201723	64-bit of data in single instance

Consider a password and self-mutation rules for the considered password as explained below:

**Password: Pooja@25**

P-80(Ascii)-1010000(decimal)

P is the first character of the password; the value of P in ASCII is 80 and the value 80 is converted to binary bit 1010000 using decimal to binary conversion method. Similarly, other characters can be converted to binary.

o-111-1101111

o-111-1101111

j-106-1101010

a-97-1100001

@-64—1000000

2-50- 10

5-53- 110101

The Table 3.3(a) shows the procedure for conversion of the password Pooja@25 to binary bits. The matrix is an 8\*8 64-bit formation. The ASCII values of Pooja@25 are converted to binary bits.

Table 3.3(a) Conversion of the password Pooja@25 to binary bits. The matrix is an 8x8 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

The self-mutation is applied to the salting key, which is generated by the operating system itself.

**Salting key: Xe92**

X-88(ASCII)- 1011000

e-101-1100101

9-57-111001

2-50- 10

The Table 3.3(b) shows the binary conversion of the salting bits where the salting key is considered as Xe92. The ASCII values of Xe92 are converted to binary bits.

Table 3.3(b) Conversion of salting key to binary bits

X	0	1	0	1	1	0	0	0
e	0	1	1	0	0	1	0	1
9	0	0	1	1	1	0	0	1
2	0	0	0	0	0	0	1	0

**Parity rules for self-key mutation:**

For 8-byte formation of matrices, a total of 56 rules are generated using permutation and combination where 28 rules are generated column-wise and the rest 28 rules are generated bitwise or row-wise. These rules can be extended with good hardware systems. To reduce complexity, the proposed research is limited to 56 rules only. The final changed matrix is randomly generated from the mutation process and cannot be identified in advance.

**Column wise parity rules for interchanging bits:**

Rule 1: 1<sup>st</sup> and 8<sup>th</sup> bit interchanging of each character.

In Table 3.4(a), the 1<sup>st</sup> bit of the password is interchanged with the last bit that is the 8<sup>th</sup> bit of the password. Hence, for character ‘P’, 1<sup>st</sup> bit ‘0’ is replaced with the 8<sup>th</sup> bit which is ‘0’ resulting in a new combination of bits ‘0101000’, though it has been observed here that both the interchangeable bits are same; same result is produced. Consider second character ‘o’ where 1<sup>st</sup> bit is ‘0’ and 8<sup>th</sup> bit is ‘1’, both are interchanged resulting in a new combination of bits ‘101101110’. In this case, the combination of bits is different from the original one.

Table 3.4(a) The 1<sup>st</sup> and 8<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0



2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

The above process follows from Rule 1 to Rule 28.

Rule 2: 1<sup>st</sup> and 7<sup>th</sup> bit interchanging of each character is shown in Table 3.4(b).

Table 3.4(b) The 1<sup>st</sup> and 7<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 3: Table 3.4(c) shows interchanging of 1<sup>st</sup> and 6<sup>th</sup> character.

Table 3.4(c) The 1<sup>st</sup> and 6<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 4: 1<sup>st</sup> and 5<sup>th</sup> bit interchanging of each character is shown in Table 3.4(d) where the first column values is interchanged with the 5<sup>th</sup> column values.

Table 3.4(d) The 1st and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 5: Table 3.4(e) shows the interchanging of 1<sup>st</sup> and 4<sup>th</sup> bit of each character where the first column values are interchanged with the 4<sup>th</sup> column values.

Table 3.4(e) The 1st and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 6: This rules interchanges the 1<sup>st</sup> and 3<sup>rd</sup> bit of each character as shown in Table 3.4(f). The first column values are replaces with the 3<sup>rd</sup> column values.

Table 3.4(f) The 1st and 3th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1

o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 7: It interchanges the 1<sup>st</sup> and 2<sup>nd</sup> bit of each character as shown in Table 3.4(g). The first column values are replaced with the 2<sup>nd</sup> column values

Table 3.4(g) The 1st and 2nd bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 8: This rule interchanges the 2<sup>nd</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(h). The 2<sup>nd</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(h) The 2<sup>nd</sup> and 8<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0

2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 9: This rule interchanges the 2<sup>nd</sup> and 7<sup>th</sup> bit of each character as shown in Table 3.4(i). The 2<sup>nd</sup> column values are replaced with 7<sup>th</sup> column values.

Table 3.4(i) The 2<sup>nd</sup> and 7<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 10: This rule interchanges the 2<sup>nd</sup> and 6<sup>th</sup> bit of each character as shown in Table 3.4(j). The 2<sup>nd</sup> column values are replaced with 6<sup>th</sup> column values.

Table 3.4(j) The 2<sup>nd</sup> and 6<sup>th</sup> bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 11: This rule interchanges the 2<sup>nd</sup> and 5<sup>th</sup> bit of each character as shown in Table 3.4(k). The 2<sup>nd</sup> column values are replaced with 5<sup>th</sup> column values.

Table 3.4(k) The 2nd and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 12: This rule interchanges the 2<sup>nd</sup> and 4<sup>th</sup> bit of each character as shown in Table 3.4(l). The 2<sup>nd</sup> column values are replaced with 4<sup>th</sup> column values.

Table 3.4(l) The 2nd and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 13: This rule interchanges the 2<sup>nd</sup> and 3<sup>rd</sup> bit of each character as shown in Table 3.4(m). The 2<sup>nd</sup> column values are replaced with 3<sup>rd</sup> column values.

Table 3.4(m) The 2nd and 3rd bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 14: This rule interchanges the 3<sup>rd</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(n). The 3<sup>rd</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(n) The 3rd and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 15: This rule interchanges the 3<sup>rd</sup> and 7<sup>th</sup> bit of each character as shown in Table 3.4(o). The 3<sup>rd</sup> column values are replaced with 7<sup>th</sup> column values.

Table 3.4(o) The 3rd and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1

j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 16: This rule interchanges the 3<sup>rd</sup> and 6<sup>th</sup> bit of each character as shown in Table 3.4(p). The 3<sup>rd</sup> column values are replaced with 6<sup>th</sup> column values.

Table 3.4(p) The 3rd and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 17: This rule interchanges the 3<sup>rd</sup> and 5<sup>th</sup> bit of each character as shown in Table 3.4(q). The 3<sup>rd</sup> column values are replaced with 5<sup>th</sup> column values.

Table 3.4(q) The 3rd and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 18: This rule interchanges the 3<sup>rd</sup> and 4<sup>th</sup> bit of each character as shown in Table 3.4(r). The 3<sup>rd</sup> column values are replaced with 4<sup>th</sup> column values.

Table 3.4(r) The 3rd and 4th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 18: This rule interchanges the 4<sup>th</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(s). The 4<sup>th</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(s) The 4th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 20: This rule interchanges the 4<sup>th</sup> and 7<sup>th</sup> bit of each character as shown in Table 3.4(t). The 4<sup>th</sup> column values are replaced with 7<sup>th</sup> column values.

Table 3.4(t) The 4th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation



P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 21: This rule interchanges the 4<sup>th</sup> and 6<sup>th</sup> bit of each character as shown in Table 3.4(u). The 4<sup>th</sup> column values are replaced with 6<sup>th</sup> column values.

Table 3.4(u) The 4th and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 22: This rule interchanges the 4<sup>th</sup> and 5<sup>th</sup> bit of each character as shown in Table 3.4(v). The 4<sup>th</sup> column values are replaced with 5<sup>th</sup> column values.

Table 3.4(v) The 4th and 5th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1

@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 23: This rule interchanges the 5<sup>th</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(w). The 5<sup>th</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(w) The 5th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 24: This rule interchanges the 5<sup>th</sup> and 7<sup>th</sup> bit of each character as shown in Table 3.4(x). The 5<sup>th</sup> column values are replaced with 7<sup>th</sup> column values.

Table 3.4(x) The 5th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 25: This rule interchanges the 5<sup>th</sup> and 6<sup>th</sup> bit of each character as shown in Table 3.4(p). The 5<sup>th</sup> column values are replaced with 6<sup>th</sup> column values.

Table 3.4(y) The 5th and 6th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 26: This rule interchanges the 6<sup>th</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(z). The 6<sup>th</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(z) The 6th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 27: This rule interchanges the 6<sup>th</sup> and 7<sup>th</sup> bit of each character as shown in Table 3.4(p). The 6<sup>th</sup> column values are replaced with 7<sup>th</sup> column values.

Table 3.4(ab) The 6th and 7th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 28: This rule interchanges the 7<sup>th</sup> and 8<sup>th</sup> bit of each character as shown in Table 3.4(p). The 7<sup>th</sup> column values are replaced with 8<sup>th</sup> column values.

Table 3.4(ac) The 7th and 8th bit interchanging of each character in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

### Parity rules for self-key mutation through Row-wise:

The rules based on interchanging a row with another row that is, bitwise operation follows here. A few bits are interchanged with another set of bits. Total 64 bits are incorporated.

Rule 29: This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 57<sup>th</sup> bit to 64<sup>th</sup> bit of each character as shown in Table 3.5(a).

In this rule, bitwise operation takes place. It shuffles the bits for one character with bits of another character. Here, character 'P' is replaced with '5', which means the bits are

interchanging. After applying the mutation, 'P' has bits '00110101' and '5' has bits '01010000'. Hence, the password is shuffled bitwise.

Table 3.5(a) The 1st bit to 8th bit interchange with 57th bit to 64th bit in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Similar process follows from Rule 30 to Rule 56.

Rule 30: This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 49<sup>th</sup> bit to 56<sup>th</sup> bit of each character as shown in Table 3.5(b).

In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with bit '2' values, which means the bits are interchanging. After applying the mutation, 'P' has bits '00000010' and '2' has bits '01010000'. Hence, the password is shuffled bitwise.

Table 3.5(b) The 1st to 8th bit interchange with 49th bit to 56th bit in 8x8 matrix vector with 64-bit formation

P	0	1	0	1	0	0	0	0
o	0	1	1	0	1	1	1	1
o	0	1	1	0	1	1	1	1
j	0	1	1	0	1	0	1	0
a	0	1	1	0	0	0	0	1
@	0	1	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0
5	0	0	1	1	0	1	0	1

Rule 31: 1<sup>st</sup> to 8<sup>th</sup> bit interchange with 41<sup>st</sup> bit to 48<sup>th</sup> bit.

This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 41<sup>st</sup> bit to 48<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with character '@' values, which means the bits are interchanging. After applying the mutation, 'P' has bits '01000000' and '@' has bits '01010000'. Hence, the password is shuffled bitwise.

Rule 32: 1<sup>st</sup> to 8<sup>th</sup> bit interchange with 33<sup>th</sup> bit to 40<sup>th</sup> bit.

This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 33<sup>rd</sup> bit to 40<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with character 'a' values, which means the bits are interchanging. After applying the mutation, 'P' has bits '01100001' and 'a' has bits '01010000'. Hence, the password is shuffled bitwise.

Rule 33: 1<sup>st</sup> to 8<sup>th</sup> bit interchange with 25<sup>th</sup> bit to 32<sup>nd</sup> bit.

This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 25<sup>th</sup> bit to 32<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with character 'j' values, which means the bits are interchanging. After applying the mutation, 'P' has bits '01101010' and 'j' has bits '01010000'. Hence, the password is shuffled bitwise.

Rule 34: 1<sup>st</sup> to 8<sup>th</sup> bit interchange with 17<sup>th</sup> bit to 24<sup>th</sup> bit.

This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 17<sup>th</sup> bit to 24<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with character 'o' values, which means the bits are interchanging. After applying the mutation, 'P' has bits '01101010' and 'o' has bits '01010000'. Hence, the password is shuffled bitwise.

Rule 35: 1<sup>st</sup> to 8<sup>th</sup> bit interchange with 9<sup>th</sup> bit to 16<sup>th</sup> bit.

This rule interchanges the 1<sup>st</sup> bit to 8<sup>th</sup> bit with 9<sup>th</sup> bit to 16<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'P' bits are replaced with character 'o' values (second row), which means the bits are interchanging. After applying the mutation, 'P' has bits '01101111' and 'o' has bits '01101010'. Hence, the password is shuffled bitwise.

Rule 36: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character '5' values, which means the bits are interchanging. After applying the mutation, 'o' has bits '00110101' and '5' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 37: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 49<sup>th</sup> to 56<sup>th</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 49<sup>th</sup> to 56<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character '2' values, which means the bits are interchanging. After applying the mutation, 'o' has bits '00000010' and '2' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 38: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 41<sup>st</sup> bit to 48<sup>th</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 41<sup>st</sup> to 48<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character '@' values, which means the bits are interchanging. After applying the mutation, 'o' has bits '01000000' and '@' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 39: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 33<sup>rd</sup> bit to 40<sup>th</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 33<sup>rd</sup> bit to 40<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character 'a' values, which means the bits are interchanging. After applying the mutation, 'o' has bits '01100001' and 'a' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 40: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 25<sup>th</sup> to 32<sup>nd</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 25<sup>th</sup> to 32<sup>nd</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character 'j' values, which means the bits are interchanging. After applying the mutation, 'o' has bits '01101010' and 'j' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 41: 9<sup>th</sup> to 16<sup>th</sup> bit interchange with 17<sup>th</sup> bit to 24<sup>th</sup> bit.

This rule interchanges the 9<sup>th</sup> to 16<sup>th</sup> bit with 17<sup>th</sup> bit to 24<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (second row) bits are replaced with character 'o' (third row) values, which means the bits are interchanging. After applying the mutation, 'o' (second row) has bits '01101111' and 'o' (third row) has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 42: 17<sup>th</sup> to 24<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 17<sup>th</sup> to 24<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (third row) bits are replaced with character '5' values, which means the

bits are interchanging. After applying the mutation, 'o' (third row) has bits '00110101' and '5' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 43: 17<sup>th</sup> to 24<sup>th</sup> bit interchange with 49<sup>th</sup> to 56<sup>th</sup> bit.

This rule interchanges the 17<sup>th</sup> to 24<sup>th</sup> bit with 49<sup>th</sup> to 56<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (third row) bits are replaced with character '2' values, which means the bits are interchanging. After applying the mutation, 'o' (third row) has bits '00000010' and '2' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 44: 17<sup>th</sup> to 24<sup>th</sup> bit interchange with 41<sup>st</sup> to 48<sup>th</sup> bit.

This rule interchanges the 17<sup>th</sup> to 24<sup>th</sup> bit with 41<sup>st</sup> to 48<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (third row) bits are replaced with character '@' values, which means the bits are interchanging. After applying the mutation, 'o' (third row) has bits '01000000' and '@' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 45: 17<sup>th</sup> to 24<sup>th</sup> bit interchange with 33<sup>rd</sup> to 40<sup>th</sup> bit.

This rule interchanges the 17<sup>th</sup> to 24<sup>th</sup> bit with 33<sup>rd</sup> to 40<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (third row) bits are replaced with character 'a' bit, which means the bits are interchanging. After applying the mutation, 'o' (third row) has bits '01100001' and 'a' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 46: 17<sup>th</sup> to 24<sup>th</sup> bit interchange with 25<sup>th</sup> to 32<sup>nd</sup> bit.

This rule interchanges the 17<sup>th</sup> to 24<sup>th</sup> bit with 25<sup>th</sup> to 32<sup>nd</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'o' (third row) bits are replaced with character 'j' bits, which means the bits are interchanging. After applying the mutation, 'o' (third row) has bits '01101010' and 'j' has bits '01101111'. Hence, the password is shuffled bitwise.

Rule 47: 25<sup>th</sup> to 32<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 25<sup>th</sup> to 32<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'j' bits are replaced with character '5' bits, which means the bits are interchanging. After applying the mutation, 'j' has bits '00110101' and '5' has bits '01101010'. Hence, the password is shuffled bitwise.

Rule 48: 25<sup>th</sup> to 32<sup>th</sup> bit interchange with 49<sup>th</sup> to 56<sup>th</sup> bit.

This rule interchanges the 25<sup>th</sup> to 32<sup>th</sup> bit with 49<sup>th</sup> to 56<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here,



character 'j' bits are replaced with character '2' bits, which means the bits are interchanging. After applying the mutation, 'j' has bits '00000010' and '2' has bits '01101010'. Hence, the password is shuffled bitwise.

Rule 49: 25<sup>th</sup> to 32<sup>th</sup> bit interchange with 41<sup>st</sup> to 48<sup>th</sup> bit.

This rule interchanges the 25<sup>th</sup> to 32<sup>th</sup> bit with 41<sup>st</sup> to 48<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'j' bits are replaced with character '@' bits, which means the bits are interchanging. After applying the mutation, 'j' has bits '01000001' and '@' has bits '01101010'. Hence, the password is shuffled bitwise.

Rule 50: 25<sup>th</sup> to 32<sup>th</sup> bit interchange with 33<sup>rd</sup> to 40<sup>th</sup> bit.

This rule interchanges the 25<sup>th</sup> to 32<sup>th</sup> bit with 33<sup>rd</sup> to 40<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'j' bits are replaced with character 'a' bits, which means the bits are interchanging. After applying the mutation, 'j' has bits '01100001' and 'a' has bits '01101010'. Hence, the password is shuffled bitwise.

Rule 51: 33<sup>rd</sup> to 40<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 33<sup>rd</sup> to 40<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'a' bits are replaced with character '5' bits, which means the bits are interchanging. After applying the mutation, 'a' has bits '00110101' and '5' has bits '01100001'. Hence, the password is shuffled bitwise.

Rule 52: 33<sup>rd</sup> to 40<sup>th</sup> bit interchange with 49<sup>th</sup> to 56<sup>th</sup> bit.

This rule interchanges the 33<sup>rd</sup> to 40<sup>th</sup> bit with 49<sup>th</sup> to 56<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'a' bits are replaced with character '2' bits, which means the bits are interchanging. After applying the mutation, 'a' has bits '00000010' and '2' has bits '01100001'. Hence, the password is shuffled bitwise.

Rule 53: 33<sup>rd</sup> to 40<sup>th</sup> bit interchange with 41<sup>st</sup> to 48<sup>th</sup> bit.

This rule interchanges the 33<sup>rd</sup> to 40<sup>th</sup> bit with 41<sup>st</sup> to 48<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character 'a' bits are replaced with character '@' bits, which means the bits are interchanging. After applying the mutation, 'a' has bits '01000000' and '5' has bits '01100001'. Hence, the password is shuffled bitwise.

Rule 54: 41<sup>st</sup> to 48<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 41<sup>st</sup> to 48<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character '@' bits are replaced with character '5' bits, which means the bits are interchanging. After applying the mutation, '@' has bits '00110101' and '5' has bits '01000000'. Hence, the password is shuffled bitwise.

Rule 55: 41<sup>st</sup> to 48<sup>th</sup> bit interchange with 49<sup>th</sup> to 56<sup>th</sup> bit.

This rule interchanges the 41<sup>st</sup> to 48<sup>th</sup> bit with 49<sup>th</sup> to 56<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character '@' bits are replaced with character '2' bits, which means the bits are interchanging. After applying the mutation, '@' has bits '00000010' and '2' has bits '01000000'. Hence, the password is shuffled bitwise.

Rule 56: 49<sup>th</sup> to 56<sup>th</sup> bit interchange with 57<sup>th</sup> to 64<sup>th</sup> bit.

This rule interchanges the 49<sup>th</sup> to 56<sup>th</sup> bit with 57<sup>th</sup> to 64<sup>th</sup> bit of each character. In this rule, shuffling of bits for one character is done with bits of another character. Here, character '2' bits are replaced with character '5' bits, which means the bits are interchanging. After applying the mutation, '2' has bits '00000010' and '5' has bits '00110101'. Hence, the password is shuffled bitwise.

A total of 56 parity rules are generated for 8-byte formation. Based on these rules, self-key mutation concept is implemented. The resultant matrix cannot be identified in advance since it is randomly generated using the key mapping bit parity rules in the mutation process and swapping pattern is anonymous to all the users whether the user is authorized or unauthorized.

### **3.9 Brief summary of this chapter**

This chapter presents the objectives of the proposed research work. The methodology of the proposed work is defined where, the working of the hybrid environment using EHC and BGV is discussed, along with CPABE parameters that are used to define customized access to different tenants. The algorithm is designed to show the step-by-step procedure of the proposed methodology. The parity rules for key mapping are defined, that are used in the attacking scenario.

## Chapter-4

### Experimental Analysis and Results

#### 4.1 Experimental setup

A real-time cloud scenario is created by designing the API using Xampp tool. Apache tomcat server acts as the web server and MySQL is used to handle the backend queries.

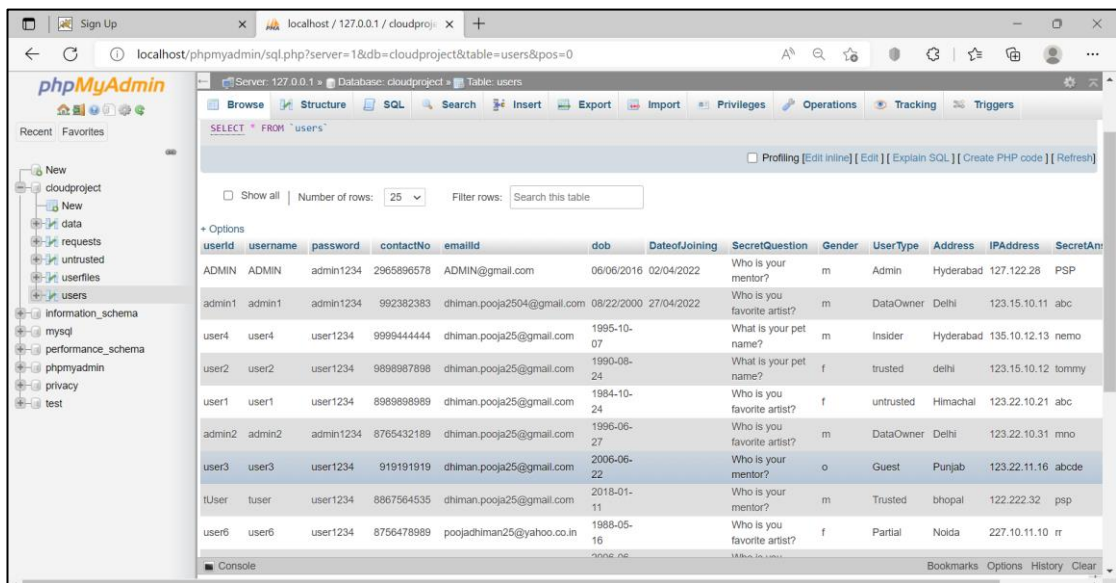
The multi-tenant cloud API is designed with two servers; the main server and the web server. The database server or main server is applicable for communication inside the organization or an enterprise; only tokens are generated for internal communication. The IP in this scenario is not visible to the attacker and hence it is considered as safest communication method while the web server is used for communication outside the organization or enterprise. The database tables created in MySQL contains the tenant information based on their role. The role is approved by ADMIN only. For accessing the files, another table is created which contains information regarding the files present in the database and the key/token generation algorithms are embedded in the code to grant access to only authenticated user. The authenticated and authorized users can only access the files, all the information is matched from the database. To show the attempt by an unauthorized user, a table named 'untrusted' is created which contains the information about the attacker such as, IP address, name, and location. After 3 invalid attempts, an email will be shared with the attacker's details to the owner of the data being accessed. The complete processing is done by integrating the algorithms in the web server interface.

The proposed model is tested for 223 end-users, 17 multi-tenants, 12 head-tenants and 7 enterprise levels.

**Case1:** Creating multiple tenants in cloud architecture.

In *figure 4.1.1*, ADMIN is the first layer of the tenant category who is responsible to approve or reject the tenant's authentication and registration. The login credentials are shared on the email id of the authenticated tenant or the user. The credentials should be modified by the user once the user logs the page, as the credentials shared by the

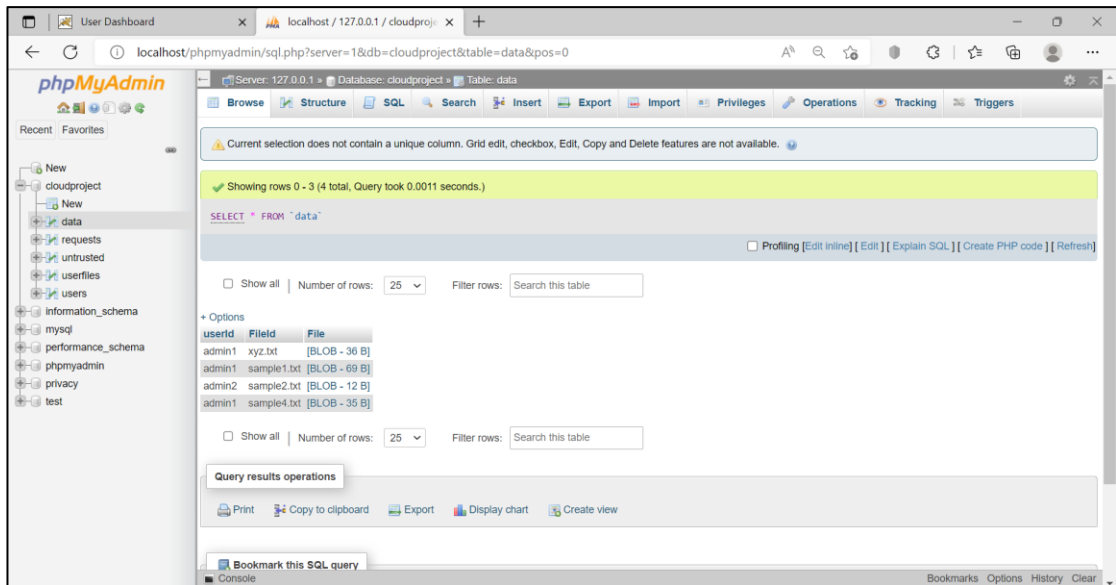
admin on the email id are the default credentials and should be changed for security reasons. Next layer is designed for data owners who are responsible to upload the data on the cloud server. Further it is divided into different user types; insider tenant, trusted tenant, partial tenant, and guest tenant. Depending on the role type, key expiration technique is followed. Hence, same token or key cannot be used to access the file for a long time or after expiry of the token or the key.



**Figure 4.1.1** Test case for creating multiple tenants in cloud architecture

**Case2:** Uploading a file on cloud server on real-time.

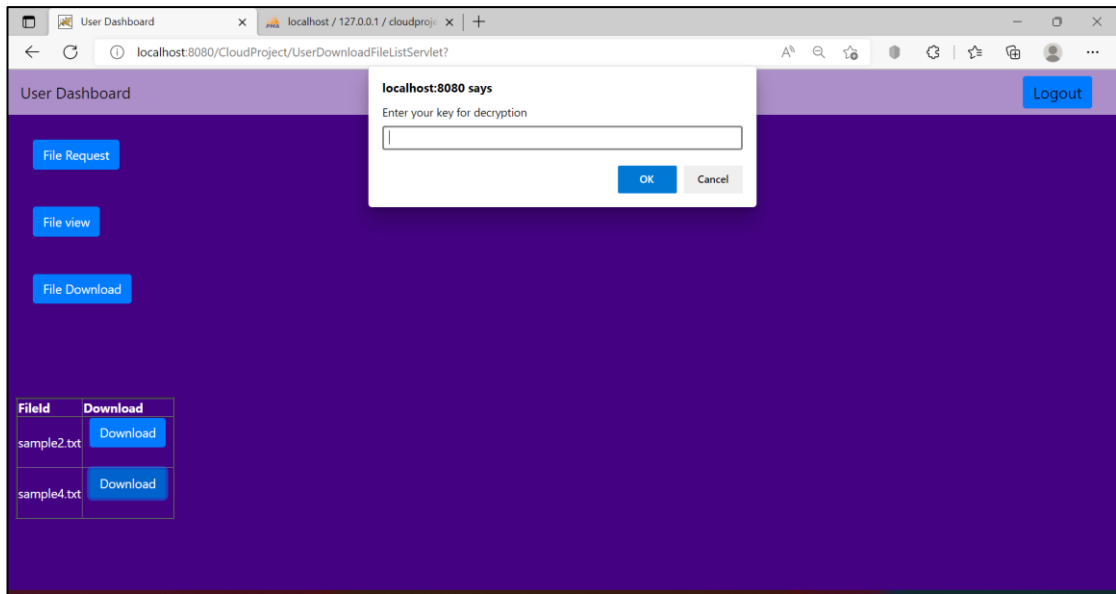
Figure 4.1.2 shows the home page of data owner named admin1. Admin1 is responsible to upload the data or files on the cloud environment. Suppose “sample4.txt” file needs to be uploaded. Clicking on “choose file” button navigates to the windows browser; the required file can be uploaded by clicking on “upload” button afterwards. Figure 4.1.2 shows the existence of the file on the cloud server.



*Figure 4.1.2 Test case for uploading a file on cloud server on real-time*

**Case3:** Request for accessing a file.

Suppose user “user4” wants to access the uploaded file “sample4.text” by “admin1”. Then “user4” first requests file access to the data owner, this request is received and approved by “admin1”, the data owner, then a key is generated using EHC algorithm since the category of “user4” is “Insider” user. For a partial user, BGV algorithm is used to generate the hybrid key. This key is sent to the user’s email id. It will expire after some time based on the user’s role category. For an Insider user, the key expiration is the maximum since it is considered the safest role as the tenant or user belongs to the organization. The decryption key received on the user’s email id is required to open the required file as shown in *Figure 4.1.3*.



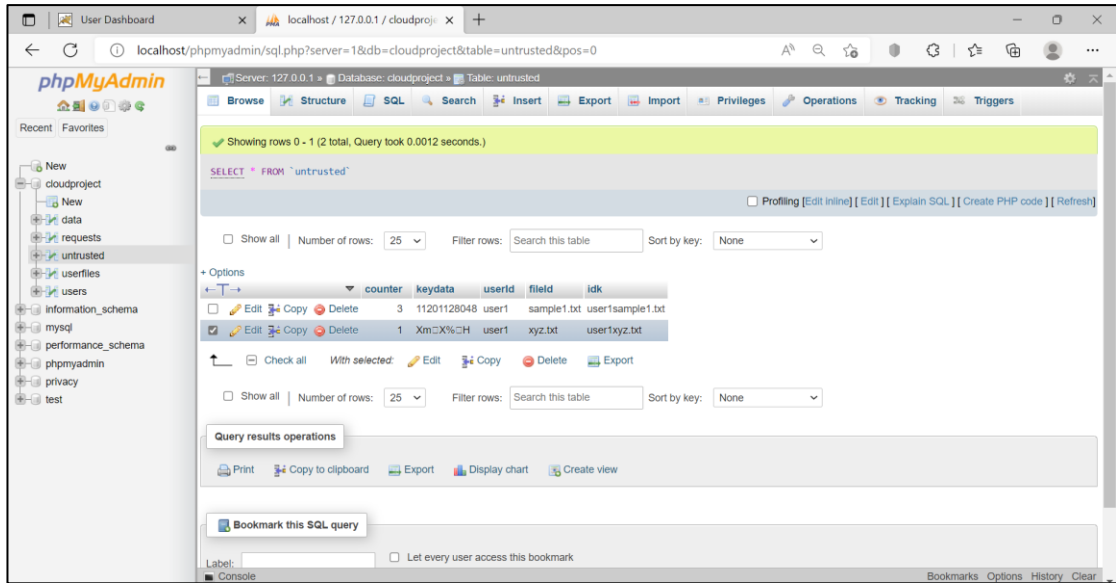
**Figure 4.1.3** Test case for Request for accessing a file.

Figure 4.1.4 shows the snip of token/key received on the email id of the user. This token or key is generated using EHC or BGV algorithm based on the category of the user.



**Figure 4.1.4** Test case for generating the token key which is received successfully on email id

**Case 4:** Self-key mutation in case of an attack. Figure 4.1.5 shows the result after the self-key mutation. A total of 56 parity rules are applied randomly. These 56 rules are defined for 8-byte bit formation or 64-bit matrix formation. In the table, “keydata” column defines the key after applying the mutation technique. The original key is mutated randomly 5 times which increases the complexity in guessing the key.



*Figure 4.1.5 Test case showing the mutated key after an attack*

## 4.2 Creating a multitenant environment on CloudSim and generating the keys using EHC algorithm

A multitenant cloud environment is designed using CloudSim simulator and Netbeans. Authentication and authorization of users is done using the credentials of the users and only after proper verification and validation of the credentials, the user can access the cloud. The bits required for PuK and PK and the time taken by these keys are evaluated. Figure 4.2(a) shows the key generation user time for EHC algorithm which is generated by implementing the EHC algorithm. The public key size is 144 bits and private key size is 152 bits and the key generation time for public key is 474.599ms and for private key, it takes 316.400ms (round off to 3 decimal places).

```

run:
Initialising...
***Welcome to simulator of Cloud***
-----
1. 1 data center created
2. 1 Hard-Drive allocated to the Data Center
-----

Cloud has been generated
-----
* * * * *
Used disk space on hdl=0.0
Available disk space on hdl=1024.0
* * * * *

Hello User
Do you want to access cloud?<y/n> : y
ENTER USER_ID : pooja
ENTER PASSWORD : 123456
Authorized User.....

Public key size : 144 bits
-----

Private key size : 152 bits
-----

Public key generation time: 474.59999999999997ms
-----

Private key generation time: 316.40000000000003ms
-----

```

Figure 4.2(a) Test case with EHC keys generation time and size

#### 4.2.1 Token generation

A token is generated for the communication inside the organization to access the file. It is sent to the registered user’s email id. FTP is used as a transmission protocol for sending tokens/keys.

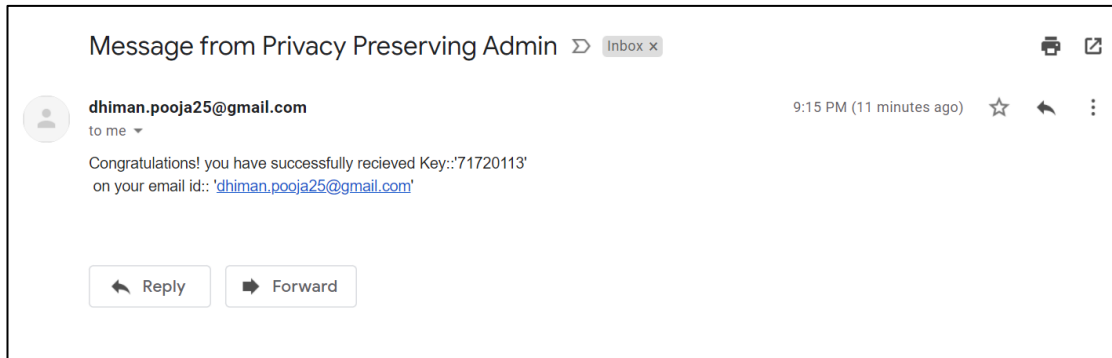
The token will expire after some specified time. After this, the session of the tenant will expire and is asked to request again to the owner for accessing that file. The tenant requests again for the file to the owner and after approval by the data owner, OTP will be received on the user’s email id and same can be used to get access to the content of the file.

#### 4.2.2 Snip for token generation on email id

The token ID is generated randomly and successfully received on the registered email id. The snip of email id is shown in *figure 4.2(b)*. An 8-bit token number is randomly generated and shared on the authenticated user’s ID. This token key will expire after



some period of time. Dummy keys are generated by the organization for its users, which should be modified by the user before logging in the system.



*Figure 4.2(b) Test case with token generation on email id*

### 4.2.3 Storage requirement by EHC

The memory requirement by EHC algorithm is very less which results in an overall inexpensive approach, though BGV requires more storage space but in the proposed approach, BGV is used only to generate hybrid key and the public key, private key and token generation are done using EHC algorithm.

The memory requirement by EHC can be evaluated by implementing EHC algorithm as shown in *Figure 4.2(c)*. The total memory required by EHC before and after key generation is evaluated as 1.11255 bits which is approximately equal to 0.0 bytes (negligible to zero). Hence, it can be concluded that to total time taken by EHC is very less.

```

Output - echalgo (run) x
run:
Total Memory Before key Generation 1.11255016E8 B
Total Memory After key Generation 1.11255016E8 B
Memory used in bytes 0.0 B
BUILD SUCCESSFUL (total time: 2 seconds)

hybridkey.java x echalgo.java x
Source History
19 long memoryUsedstart = runtime.freeMemory();
20 System.out.println("Total Memory Before key Generation "+(double)(memoryUsedstart)+" B");
21 byte[] fooBytes = p.getBytes();
22 //System.out.println("encode result:"+fooBytes.length);
23
24 BigInteger m = new BigInteger(fooBytes);
25
26 cipherText c = new cipherText(genModulus.Encode(keypair.getPublicKey(),m, keypair.getPublicKeyparam())
27 long memoryUsedstop = runtime.freeMemory();
28 System.out.println("Total Memory After key Generation "+(double)(memoryUsedstop)+" B");
29 long memoryUsed = memoryUsedstart - memoryUsedstop;
30 System.out.println("Memory used in bytes "+(double)(memoryUsed)+" B");
31 BigInteger decode_res = genModulus.Decode(keypair.getPrivateKey(),c.getCipherText(),keypair.getPublic
32 byte[] decodedval = decode_res.toByteArray();
33
34 String s = new String(decodedval);
35 //System.out.println("decode result:"+s);
36

```

Figure 4.2(c) Memory required by EHC

### 4.3 Test case analysis

The proposed EHC-BGV encryption hybrid model is much better in terms of providing security and privacy with the implications of transmission rate of data, secure tokens number and keys generations, application areas along with the implicated environment-internal enterprise like as local-global-local tenant and implicated environment-external enterprise like as external enterprise.

#### 4.3.1 Implications of EHC algorithm based key generation time, encryption time and decryption time

The key generation time, encryption time and decryption time of EHC algorithm is evaluated by implementing the EHC algorithm as shown in the *Figure 4.3*. The file size varies from 24 bits to 8248 bits. It can be observed that the encryption, and decryption time of EHC is better if compared with the past proposed algorithms like RSA, AHEE etc. Some of the algorithms are compared and shown in the upcoming sections.

```

Hello User
Do you want to access cloud?<y/n> : y
ENTER USER_ID : pooja
ENTER PASSWORD : 123456
Authorized User.....
Size of Ciphertext for PT of Size 24 bit is 58bits
Time in Key Generation of 24bits in (ns)1257100
Time in Encryption Generation of 24bits in (ns)640800
Time in Decryption Generation of 24bits in (ns)710700
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 72 bit is 145bits
Time in Key Generation of 72bits in (ns)1257100
Time in Encryption Generation of 72bits in (ns)938000
Time in Decryption Generation of 72bits in (ns)1378200
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 128 bit is 269bits
Time in Key Generation of 128bits in (ns)1257100
Time in Encryption Generation of 128bits in (ns)1324900
Time in Decryption Generation of 128bits in (ns)2729300
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 256 bit is 257bits
Time in Key Generation of 256bits in (ns)1257100
Time in Encryption Generation of 256bits in (ns)1705800
Time in Decryption Generation of 256bits in (ns)4001100
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 1024 bit is 270bits
Time in Key Generation of 1024bits in (ns)1257100
Time in Encryption Generation of 1024bits in (ns)2144100
Time in Decryption Generation of 1024bits in (ns)5408500
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 2048 bit is 249bits
Time in Key Generation of 2048bits in (ns)1257100
Time in Encryption Generation of 2048bits in (ns)2653600
Time in Decryption Generation of 2048bits in (ns)6742800
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 4096 bit is 250bits
Time in Key Generation of 4096bits in (ns)1257100
Time in Encryption Generation of 4096bits in (ns)3183400
Time in Decryption Generation of 4096bits in (ns)7643900
Memory used in percentage (ENC+DEC): 0.46773624%
Size of Ciphertext for PT of Size 8248 bit is 255bits
Time in Key Generation of 8248bits in (ns)1257100
Time in Encryption Generation of 8248bits in (ns)3778100
Time in Decryption Generation of 8248bits in (ns)8920800
Memory used in percentage (ENC+DEC): 0.46773624%
Starting CloudSim version 3.0
MyDC1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
MyDC1 is shutting down...
Broker is shutting down...
Simulation completed.
***UPDATED HARD DISK SPACE ***
Used disk space on hdl=0.0
BUILD SUCCESSFUL (total time: 9 seconds)

```

*Figure 4.3 Implications of EHC algorithm based key generation time, encryption time, decryption time and memory consumed*

### **4.3.2 Implications of Hybrid EHC-BGV Homomorphic based key generation time, encryption time and decryption time**

The key generation time, encryption time and decryption time of hybrid approach has shown as in the *Figure 4.4* by running the implementation code as follows. It can be noted that the time taken by EHC is more in case of encrypting and decrypting a file when compared with the results obtained from the hybrid model that is, when EHC is integrated with BGC, it gives better results.

```

Hello User
Do you want to access cloud?<y/n> : y
ENTER USER_ID : pooja
ENTER PASSWORD : 123456
Authorized User.....
Size of Ciphertext for PT of Size 24 bit is 24bits
Time in key Generation of 24bits in (ns)298400
Time in Encryption Generation of 24bits in (ns)28400
Time in Decryption Generation of 24bits in (ns)39300
Memory used in percentage (ENC+DEC): 0.18104689%
Size of Ciphertext for PT of Size 72 bit is 56bits
Time in key Generation of 72bits in (ns)128100
Time in Encryption Generation of 72bits in (ns)71900
Time in Decryption Generation of 72bits in (ns)111200
Memory used in percentage (ENC+DEC): 0.18104689%
Size of Ciphertext for PT of Size 128 bit is 96bits
Time in key Generation of 128bits in (ns)125500
Time in Encryption Generation of 128bits in (ns)107300
Time in Decryption Generation of 128bits in (ns)274800
Memory used in percentage (ENC+DEC): 0.21723808%
Size of Ciphertext for PT of Size 256 bit is 192bits
Time in key Generation of 256bits in (ns)210300
Time in Encryption Generation of 256bits in (ns)149200
Time in Decryption Generation of 256bits in (ns)517500
Memory used in percentage (ENC+DEC): 0.2896332%
Size of Ciphertext for PT of Size 1024 bit is 960bits
Time in key Generation of 1024bits in (ns)142300
Time in Encryption Generation of 1024bits in (ns)339500
Time in Decryption Generation of 1024bits in (ns)747600
Memory used in percentage (ENC+DEC): 0.57918954%
Size of Ciphertext for PT of Size 2048 bit is 1984bits
Time in key Generation of 2048bits in (ns)119900
Time in Encryption Generation of 2048bits in (ns)755200
Time in Decryption Generation of 2048bits in (ns)1235800
Memory used in percentage (ENC+DEC): 1.1221467%
Size of Ciphertext for PT of Size 4096 bit is 4032bits
Time in key Generation of 4096bits in (ns)105600
Time in Encryption Generation of 4096bits in (ns)1088000
Time in Decryption Generation of 4096bits in (ns)1519400
Memory used in percentage (ENC+DEC): 0.25742748%
Size of Ciphertext for PT of Size 8248 bit is 8184bits
Time in key Generation of 8248bits in (ns)110700
Time in Encryption Generation of 8248bits in (ns)1845700
Time in Decryption Generation of 8248bits in (ns)1978500
Memory used in percentage (ENC+DEC): 0.30209088%
Starting CloudSim version 3.0
MyDC1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
MyDC1 is shutting down...
Broker is shutting down...
Simulation completed.
***UPDATED HARD DISK SPACE ***
Used disk space on hdl=0.0
BUILD SUCCESSFUL (total time: 9 seconds)

```

**Figure 4.4 Implications of Hybrid EHC-BGV Homomorphic based key generation time, encryption time, decryption time and memory consumed**

### 4.3.3 Implications of Hybrid EHC-BGV Homomorphic based automated key filter scenario with bits-shuffling mechanism

The bits are shuffled based on the 8-byte formation matrix. A total of 56 parity rules are applied for key mutation concept. The result is shown in the *Figure 4.5* where the original key Pooja@25 is shuffled and mutation technique is applied which returns P2oja@05. Here, it can be observed that 2 bits are interchanged; the second bit and the seventh bit. The production rule is randomly selected from 56 rules.

```
2. 1 Hard-Drive allocated to the Data Center
-----
Cloud has been generated
-----
* * * * *
Used disk space on hdl=0.0
Available disk space on hdl=1024.0
* * * * *

Hello User
Do you want to access cloud?<y/n> : y
ENTER USER_ID : pooja
ENTER PASSWORD : 123456
Authorized User.....
Starting CloudSim version 3.0
MyDC1 is starting...
Broker is starting...
Entities started.
0.0: Broker: Cloud Resource List received with 1 resource(s)
Simulation: No more future events
CloudInformationService: Notify all CloudSim entities for shutting down.
MyDC1 is shutting down...
Broker is shutting down...
Simulation completed.
***UPDATED HARD DISK SPACE ***
Used disk space on hdl=0.0
Randomly selected Rule No. 37
Time in Key Shuffle (in ns) 1100
Final result after conversion in binary
1010000
110010
1101111
1101010
1100001
1000000
1101111
110101
Final Result after conversion in string form
P
2
o
j
a
@
o
5
BUILD SUCCESSFUL (total time: 8 seconds)
```

*Figure 4.5 Implications of Hybrid EHC-BGV Homomorphic based automated key filter scenario with bits-shuffling mechanism based on the 8-byte formation matrix by 56 parity rules key mutations*

#### 4.3.4 Key generation time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size

As shown in *Table 4.1*, the key generation time in hybrid is less as compared to the EHC algorithm since a hybrid model takes advantages of the BGV and EHC algorithms both and the weaknesses of individual algorithms BGV and EHC is bypassed by using the hybrid approach.

Table 4.1 The key generation time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key

Key Size	EHC (in ms)	Hybrid (in ms)
24 bits	0.509	0.5025
72 bits	5.09	0.1954
128 bits	5.09	0.1766
256 bits	5.09	0.2261
1024 bits	5.09	0.233
2048 bits	5.09	0.1211
4096 bits	5.09	0.2239
8248 bits	5.09	0.1099

#### 4.3.5 Encryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size

The encryption time in hybrid is very less as compared with EHC algorithm as the hybrid model combines the qualities of two algorithms that is, BGV and EHC. The time required to encrypt data is less as compared with the EHC algorithm alone as shown in the *Table 4.2*.

Table 4.2 The encryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key

Plain Text Size	EHC (in ms)	Hybrid (in ms)
24 bits	2.3233	0.0355
72 bits	3.1756	0.0863
128 bits	3.8193	0.113

256 bits	4.9124	0.1493
1024 bits	6.0552	0.3443
2048 bits	7.2221	0.7684
4096 bits	7.8624	1.1374
8248 bits	8.7277	1.8491

#### 4.3.6 Decryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key size

The use of hybrid approach that integrates two FHE algorithms that is, EHC and BGC gives better results as compared with EHC algorithm alone. As seen in *Table 4.3*, the decryption time in hybrid is very less as compared with EHC algorithm.

Table 4.3 The decryption time variations among EHC algorithm and Hybrid EHC-BGV on 8-byte key

Plain Text Size	EHC (in ms)	Hybrid (in ms)
24 bits	2.1533	0.0525
72 bits	4.1264	0.1582
128 bits	6.2627	0.3305
256 bits	8.6085	0.6308
1024 bits	11.4445	0.8546
2048 bits	14.0209	2.407
4096 bits	16.8254	2.6386
8248 bits	19.4799	3.3513

#### 4.3.7 Description of Comparative analysis of proposed hybrid EHC-BGV approach with existing approaches

The novel concept in our proposed model is based on using the Hybrid EHC-BGV Homomorphic based automated key filter bit-mapping key mutation (KM) approach for Secure Data Access Control in Distributed Enterprise Multitenant. The Hybrid EHC-BGV Homomorphic based Automated KF-BM-KM is implemented only in case of an attack. When an attacker tries to intercept the transmission channel, self-key



mutation takes place and shuffling of bits is done which is based on the production rules.

In Table 4.4, a comparative description of the past proposed approaches with the proposed model is shown. It can be observed that the proposed model provides up to 96% success rate. The 96% success rate is calculated by considering the average of encryption, decryption, and key generation time of the past proposed and the proposed model.

Table 4.4 Description of Comparative analysis of proposed hybrid EHC-BGV approach with existing approaches

<b>Author name</b>	<b>Application area</b>	<b>Mode of security</b>	<b>Homomorphic encryption technique</b>	<b>Implicated Environment-Internal Enterprise</b>	<b>Implicated Environment-External Enterprise</b>	<b>Number of Tokens and Keys</b>	<b>Success rate</b>
Gorti VNKV Subba Rao et.al.[17]	Mobile Ad Hoc Networks	Message Transmission	EHC	Local Tenant	Global Tenant	4 (1 public, 3 private keys)	100% only if there are a greater number of active nodes in each group of networks.
S.V.Suriya prasad et.al. [13]	Cloud Security	Data Transmission	BGV, EBGV (Enhanced BGV)	Local Tenant	Global Tenant	2 (1 public, 1 private key)	Partially Executed
Liang Chen et.al. [14]	Cloud Security	Data Transmission	NEHE	Local Tenant	Local Tenant	4 (1 public, 3 private keys)	Partially Executed
Pooja and Santosh Kumar Henge	Network and Cloud Security	Data storage and Transmission	EHC-BGV	Local – Global –Local Tenant	External Enterprise	3 tokens PuK, HK, PK (6 keys)	96%

The complexity in securing the data on cloud for the proposed model is found in identifying the type of the tenant as the implementation of the model is based on the role of the tenant. Hence, proper evaluation of the tenant's role must be identified. The

evaluation of attacker's IP address and data site and key generation is a tedious task. Further the parameters used for authentication and authorization includes password, OTP, OOTP, token, and key expiration time. Here, OOTP is On-demand OTP that is generated only case of an attack and token/key is expired based on the category or role of the tenant/user. For the proposed multitenant cloud architecture, 2 hours token/key expiration is set for an insider user and 24 hours for a partial user and 10 minutes for anonymous user. While trusted category requires extra layer of authentication using OTP that is received on email id and to access a file, password is sent to the email id again. Hence, multi-factor authentication technique is incorporated in the proposed research model.

The proposed Hybrid EHC-BGV Homomorphic based automated key filter bit-mapping key mutation (KM) approach is more efficient in terms of data privacy and security.

#### **4.3.8 Comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with key size of 8 bytes**

The *Table 4.5* describes a comparative study of the proposed hybrid model with the existing approaches by considering the cipher-texts which are used for data transmission through the cloud enterprise-level servers. It can be observed that there is a constant key generation time for proposed EHC algorithm. The proposed EHC algorithm is tested on the proposed model which is integrated with different tenant roles, different enterprises, and different users. EHC algorithm is tested for 256-bit size only because if the key size increases EHC is taking a lot of time in key generation. It is one of the major complexities during the implementation process; the results are implicated considering the key size.

*Figure 4.6* shows the graphical representation of the results evaluated in the *Table 4.5*. It can be seen directly that the ciphertext size in the proposed model is less as compared to the past proposed algorithms.

Table 4.5 Comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with key size of 8 bytes

Plain text size in bits	RSA [3]	AHEE [3]	Proposed EHC Model	Proposed Hybrid EHC-BGV Homomorphic based automated KF-BM-KM approach
24 bits	30	28	24	24
72 bits	80	78	56	56
128 bits	137	133	64	96
256 bits	272	263	64	192
1024 bits	1033	1027	64	960
2048 bits	2057	2055	64	1984
4096 bits	4123	4115	64	4032
8248 bits	8313	8295	64	8184

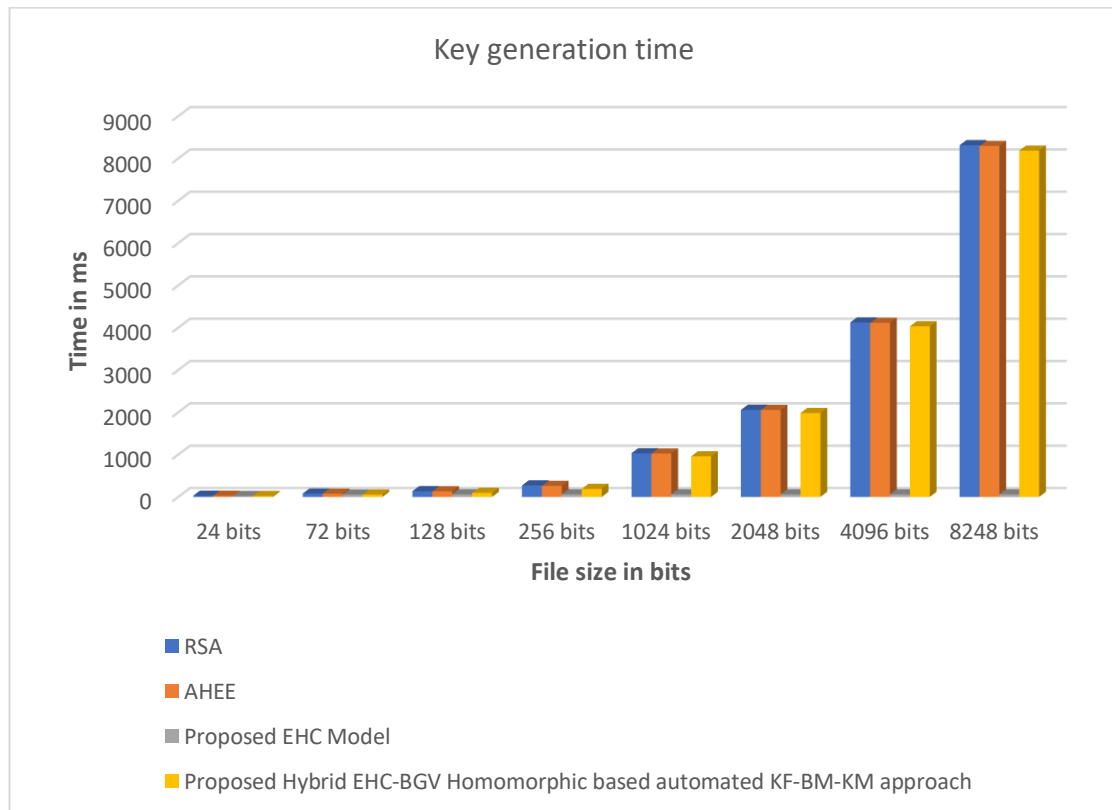


Figure 4.6 Test case with comparison of considered size of the cipher-text in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size

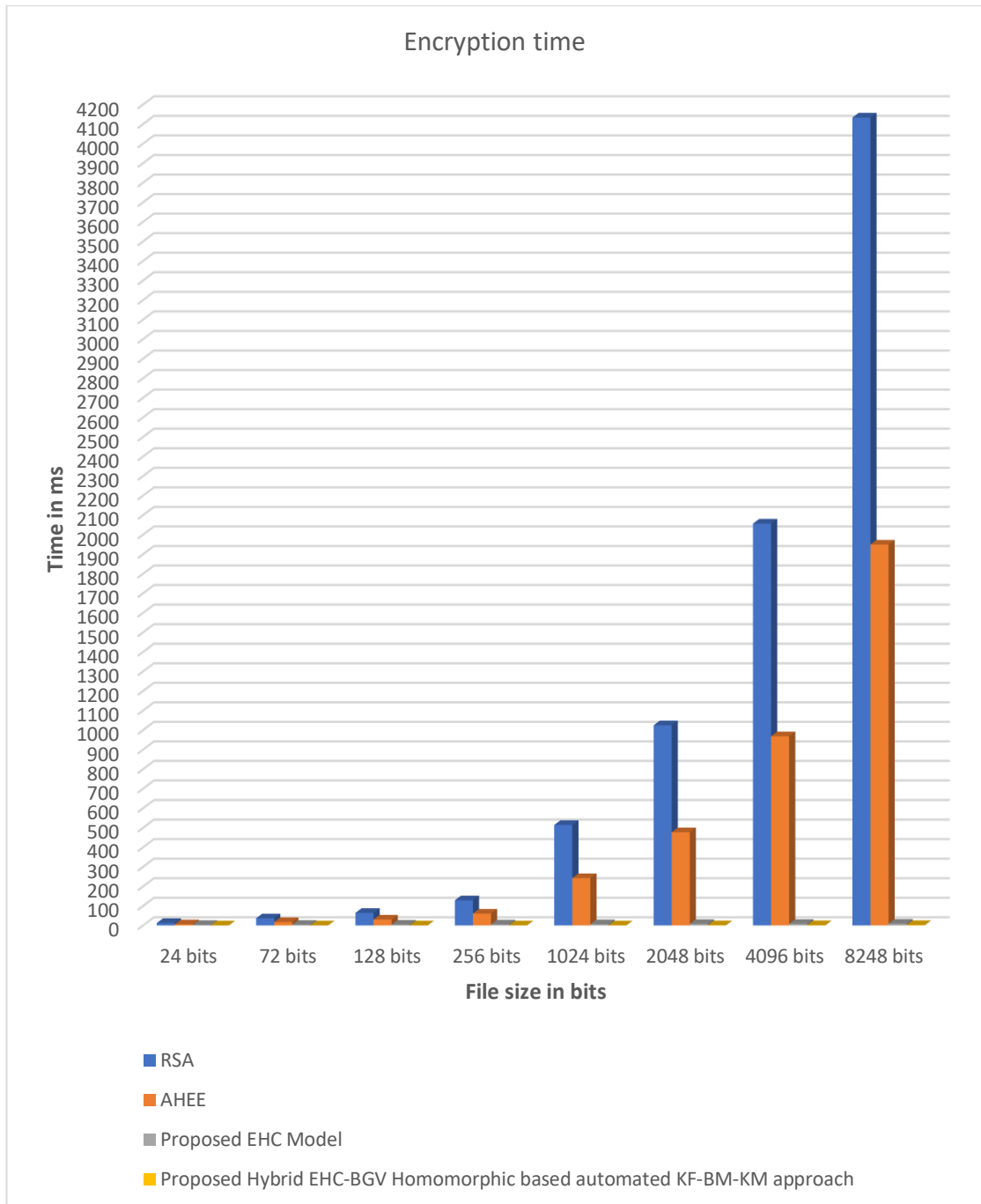
#### 4.3.9 Comparison of considered time for the implication of encryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with key size of 8 bytes

The comparison study of existing approaches along with the proposed hybrid approach by considering the encryption time taken for transmitting the data is shown as in the *Table 4.6*. The encryption time in hybrid is very less as compared with EHC algorithm for file size from 24 bits to 8248 bits. For 8248 bits file size, the time taken in the encryption process is 1.8491ns which is very less as compared to RSA, AHEE and proposed EHC algorithm. The hybrid model combines the features of both the algorithms which result in better approach.

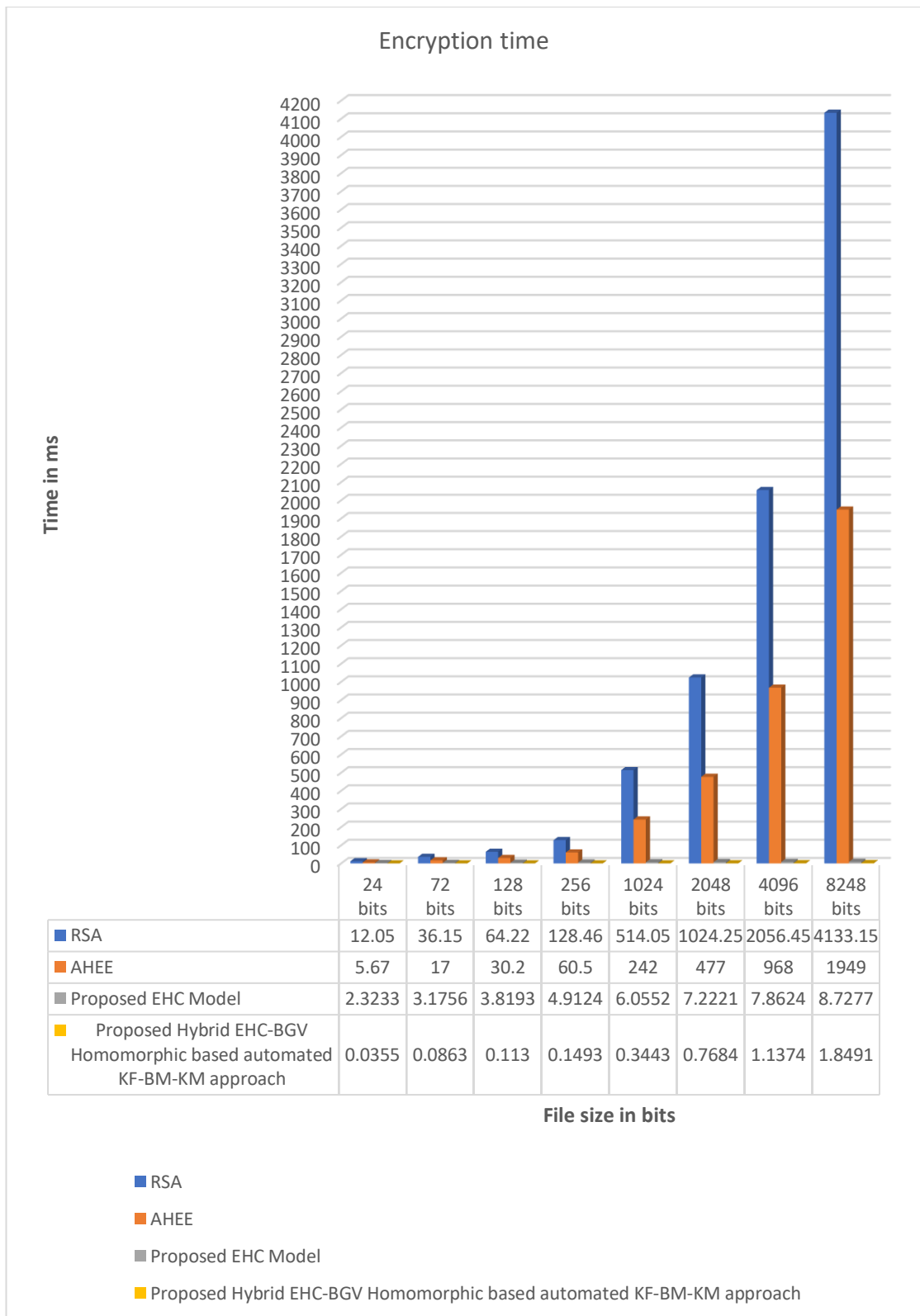
Table 4.6 Comparison of considered time for the implication of encryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with key size of 8 bytes

Plain text size in bits	RSA [3]	AHEE [3]	Proposed EHC Model	Proposed Hybrid EHC-BGV Homomorphic based automated KF-BM-KM approach
24 bits	12.05	5.67	2.3233	0.0355
72 bits	36.15	17	3.1756	0.0863
128 bits	64.22	30.2	3.8193	0.113
256 bits	128.46	60.5	4.9124	0.1493
1024 bits	514.05	242	6.0552	0.3443
2048 bits	1024.25	477	7.2221	0.7684
4096 bits	2056.45	968	7.8624	1.1374
8248 bits	4133.15	1949	8.7277	1.8491

*Figure 4.7* shows the graphical representation of the results evaluated for encryption time for 8 bytes. It can be seen directly that the encryption time in the proposed hybrid model is less as compared to the past proposed algorithms. *Figure 4.8* shows the bitwise comparison of the proposed algorithm with the past proposed algorithms.



**Figure 4.7** Test case with comparison of considered time for implication of encryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size



**Figure 4.8 Bit wise comparison of encryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size**

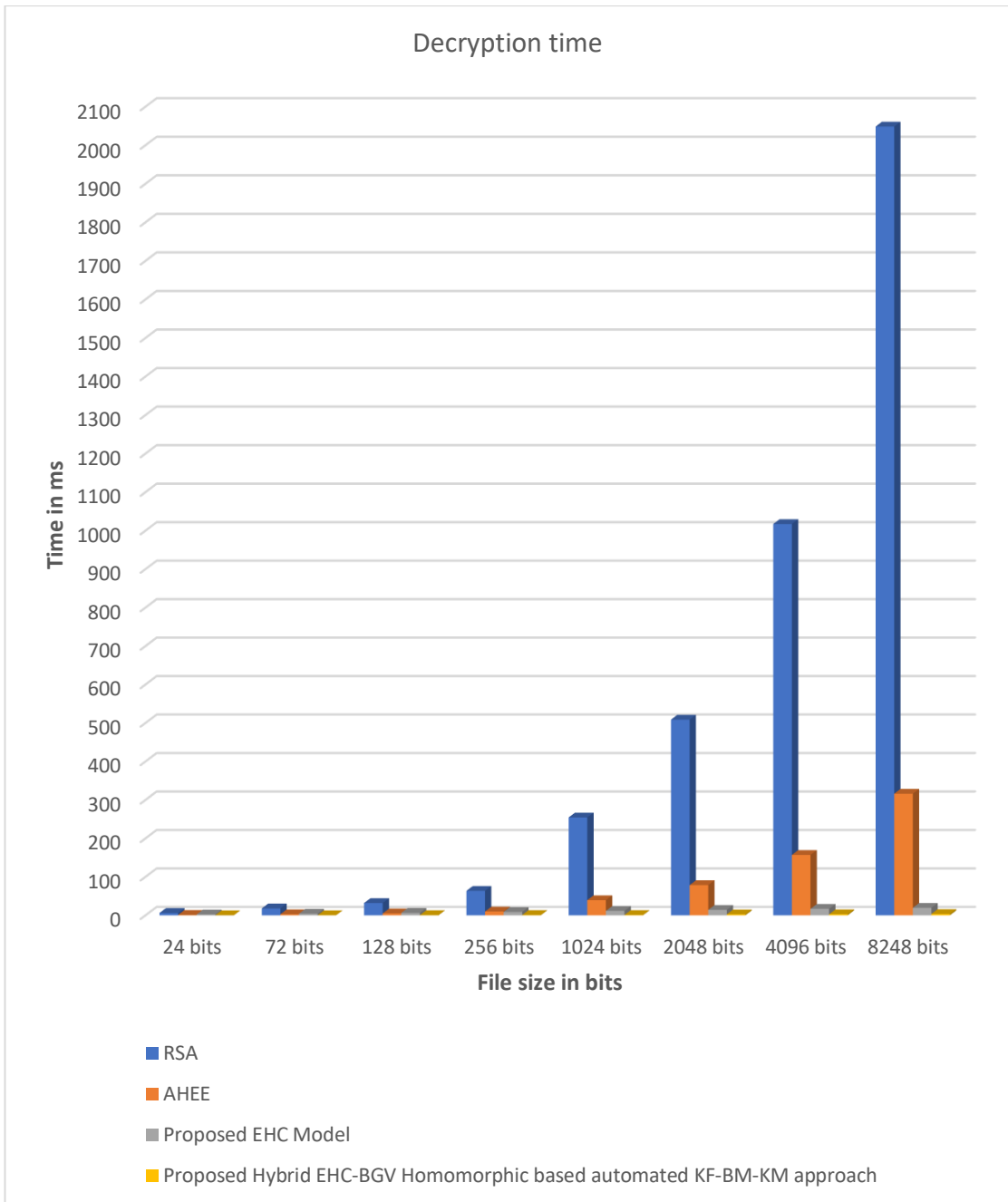
#### 4.3.10 Comparison of considered time for the implication of decryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with key size of 8 bytes

The existing approaches are compared with the proposed hybrid approach by taking into the consideration decryption time of data transmission as shown in the *Table 4.7*. The decryption time in hybrid is very less as compared with EHC algorithm for file size from 24 bits to 8248 bits. For 8248 bits file size, the time taken in the decryption process is 3.3513ns which is very less as compared to RSA, AHEE and proposed EHC algorithm.

*Figure 4.9* shows the graphical representation of the results evaluated for decryption time for 8 bytes from the *Table 4.7*. It can be seen directly that the decryption time in the proposed hybrid model is less as compared to the past proposed algorithms.

Table 4.7 Comparison of considered time for the implication of decryption-time in existing approaches along with the proposed EHC Model and hybrid EHC-BGV approach with 8 byte key size

Plain text size in bits	RSA [3]	AHEE [3]	Proposed EHC Model	Proposed Hybrid EHC-BGV Homomorphic based automated KF-BM-KM approach
24 bits	5.96	0.92	2.1533	0.0525
72 bits	17.88	2.76	4.1264	0.1582
128 bits	31.78	4.9	6.2627	0.3305
256 bits	63.57	9.8	8.6085	0.6308
1024 bits	254.31	39.3	11.4445	0.8546
2048 bits	508.59	78.5	14.0209	2.407
4096 bits	1017.17	157	16.8254	2.6386
8248 bits	2048.27	316	19.4799	3.3513



**Figure 4.9 Comparison of considered time for implication of decryption-time in existing approaches along with the proposed EHC model and hybrid EHC-BGV approach with 8-byte key size**

Hence, the proposed hybrid approach is better in terms of key generation time, encryption time and decryption time. Hence, it can be concluded that the proposed model is better in terms of providing better security and time required for generating



the keys, encryption time and decryption time is very less as compared to the EHC algorithm.

#### **4.4 Conclusion**

The proposed model uses a hybrid approach embedded with mutation technique to secure the cloud data. The hybrid technique is using EHC and BGV algorithms. Both fully homomorphic encryption schemes are compatible with each other as they overcome each other's cons. Different access methods are used for different categories of the tenant; it is based on the role of the user or tenant. Since all the services are not required by the tenants, only a few tenants require all the services; hence data is provided based on the role and category of the tenant. Based on the role of the user, an appropriate FHE scheme is applied.

If the user's role is Insider, Trusted Outsider or Anonymous/Guest user, then EHC algorithm is used. While BGV algorithm is used only in case the user is a Partial user. For an Insider user, no keys are generated; only tokens are used internally since risk of breaching data is very low inside the organization. These tokens are generated using EHC algorithm. For Outsider category, user is divided into two sub-categories; Trusted Outsider and Un-trusted Outsider. For a Trusted Outsider and a guest user, EHC is used to generate the public and private keys. Guest user is considered as more prone to breaches since risk is high in this case. For an Untrusted Partial category, hybrid key is generated using a BGV algorithm.

For handling the attacking scenario, a dictionary is maintained with a list of authorized users that contains information like IP address, name, and location of the user. If any of the data is found mismatched while requesting access to the data, it is considered as an unauthorized access. In this case, a subsequent action takes place which implements the self-key mutation concept.

The self-key mutation is applied on 8-byte or 64-bits formation of keys. A total of 56 parity rules are designed for self-key mutation implementation. Keys are shuffled based on the random selection of these parity rules.

The proposed approach has tested on various distributed cloud servers with 223 end-users by the integration of seventeen multitenant, twelve head-tenants, and seven enterprise levels.

It is found that the proposed hybrid approach is more secure than the current encryption models for protecting the data on the cloud since it is IND-CCA secure and the time taken for various operations is comparatively less. It is an inexpensive approach as storage requirement and power consumption is low. Moreover, the bit parity mapping in the hybrid approach is integrated to add a multi-factor authentication technique. Hence, it can be concluded that the proposed model is better in terms of maintaining privacy and security.

The proposed blended model is efficient to prevent the data from the ciphertext attacks and achieved a success rate of 96 percent for the communication between the multitenants that are based on the user-role-user type of enterprise cloud servers.

#### **4.5 Brief summary of this chapter**

This chapter implements the possible test cases for the proposed research methodology using XAMPP tool. The proposed hybrid approach is compared with the past proposed algorithms. It is found from the results that the proposed hybrid model using EHC and BGV that is integrated with key mutation mapping technique takes less time for encryption, decryption, and key generation process. Since, auto tuning method for keys mutation is used in self-key mutation technique, the proposed work is found to be fulfilling its objective of securing the cloud with less complexity. Based on the tenant's role, the scenario changes which is dynamically designed and implemented with more efficiency.

## Chapter-5

### Recommendations and Future work

This thesis presents a more secured way to protect the data on the cloud. Security and privacy constraints are fulfilled in the proposed study. Various access control methods are discussed along with the fully homomorphic encryption algorithm. The proposed model is designed using a hybrid FHE approach integrated with self-mutation technique.

#### 5.1 Summary

The research aims at improving security and privacy against the most vulnerable threats of cloud computing. The proposed model uses FHE algorithms, EHC and BGV for encrypting the data on cloud and access control mechanism is integrated with key mutation technique. The access is given based on the role of the user or tenant and public, private keys, and hybrid keys are generated according to the tenant's access permissions. Self-key mutation occurs in case on an attacking scenario.

The objectives of the proposed model are achieved successfully following the proposed research methodology.

**Objective 1:** To analyse existing Homomorphic techniques in multi-tenant environment.

Literature review is used for the analyses and evaluation of the past proposed HE algorithms. A combination of EHC and BGV FHE algorithms is found to be suitable for the proposed research.

**Objective 2:** To design multi-tenant access control logs in Homomorphic encryption technique.

A multitenant customized cloud environment is designed with access control based on the role of the tenant. Keys are generated on the basis of the role of the user or tenant. Cloud Simulation Tool with Netbeans is used for the implementation. Xampp is used to create web pages. Apache is used as a web server and MySQL is used as a database

server. A multi-tenant cloud environment is designed that serves different access mechanisms based on the user's role. Mapping of the tokens between tenants is based on session, and token management (initiation time, start time, end time). A Multi-factor authentication (Email, owner approval, random keys/tokens) technique is introduced. A hidden key OOTP is generated for the attacking scenario.

**Objective 3:** To develop Homomorphic encryption technique in multi-tenant cloud environment.

A secured multi-tenant cloud environment is developed using a hybrid Homomorphic approach with key filtration technique integrated in the proposed model. Cloud Simulation Tool with Netbeans is used for the implementation. Xampp is used to create web pages, Apache is used as a web server and MySQL is used as a database server. Key/token is generated using EHC and BGV FHE algorithms. EHC is used to generate the token, public key, and private key, and BGV is used to generate a hybrid key based on the user's role. Self-key mutation technique is used for the attacking scenario. A total of 56 parity rules are integrated for parity mapping.

**Objective 4:** To test and validate the developed technique.

The proposed model is compared with current methodologies and techniques, considering the parameters such as Key generation time, Encryption time, Decryption time, and Computational Complexity. It is found that the proposed methodology is secure under CCA. The time taken in encryption, decryption and key generation is very less. Success rate of the proposed model is 96%.

## 5.2 Implications

The future work can be done to reduce the computational complexity by increasing the key size. In the proposed model, the key size is limited to 8248 bits and increasing the key size increases the complexity. It can be improved by considering the high-performance devices with 128 GB RAM and specialized hardware configuration.

### **5.3 Limitations and Future scope**

- The proposed methodology is designed for 8 bytes formation of bits integrated with self-key mutation technique. In future, the number of bytes can be increased to check the performance of the proposed hybrid model.
- The EHC algorithm is tested for 256 bits fixed key size since in the proposed model, some complications were seen to handle the size more than 256 bits. It took more time in generating the keys for larger key size.
- Fibre optics can be used for further research for improved performance and increased key size. As increasing key size results in increased computational complexity due to requirement of more resources. The high-performance systems are required to perform complex computations.
- Proper evaluation can be made for the future threats. The upcoming breaches and possible attacks can be examined further to check the validity of the proposed model.
- The proposed model can be integrated with the Blockchain technique[92] and performance can be evaluated for any improvement in terms of security and in reduction the total time required in generating the keys, encryption and decryption time.

## Bibliography

- [1] N. Veeraragavan, "Design and Implementation of Authentication as a Service (Aaas) in Windows Azure Cloud Platform," *J. Phys. Conf. Ser.*, vol. 1142, no. 1, pp. 0–8, 2018, doi: 10.1088/1742-6596/1142/1/012016.
- [2] D. Pooja, "Cloud Computing - Overview and its Challenges," *Res. Rev. Int. J. Multidiscip.*, vol. 04, no. 03, pp. 499–501, 2019.
- [3] R. Kanagavalli and S. Vagdevi, "Secured Data Storage in Cloud Using Homomorphic Encryption," *Int. J. Cloud Comput. Serv. Archit.*, vol. 9, no. 4, pp. 1–11, 2019, doi: 10.5121/ijccsa.2019.9401.
- [4] I. Ahmad and A. Khandekar, "Homomorphic Encryption Method Applied to Cloud Computing," vol. 4, no. 15, pp. 1519–1530, 2014.
- [5] B. Chen and N. A. Zhao, "FULLY HOMOMORPHIC ENCRYPTION APPLICATION IN CLOUD COMPUTING," 2014.
- [6] J. Yang, M. Fan, G. Wang, and Z. Kong, "Simulation Study Based on Somewhat Homomorphic Encryption," *J. Comput. Commun.*, vol. 2014, no. January, pp. 109–111, 2014.
- [7] N. Jain, "Implementation and Analysis of Homomorphic Encryption Schemes," *Int. J. Cryptogr. Inf. Secur.*, vol. 2, no. 2, pp. 27–44, 2012, doi: 10.5121/ijcis.2012.2203.
- [8] P. V. Parmar, S. B. Padhar, S. N. Patel, N. I. Bhatt, and R. H. Jhaveri, "Survey of Various Homomorphic Encryption algorithms and Schemes," *Int. J. Comput. Appl.*, vol. 91, no. 8, pp. 26–32, 2014, doi: 10.5120/15902-5081.
- [9] T. Oladunni and S. Sharma, "Homomorphic Encryption and Data Security in the Cloud," no. October, 2019.
- [10] O. Ethelbert, F. F. Moghaddam, P. Wieder, and R. Yahyapour, "A JSON Token-Based Authentication and Access Management Schema for Cloud SaaS Applications," 2018.
- [11] B. S. Al-attab, "Authentication Technique by Using USB Token in Cloud Computing," no. February 2016, pp. 1–4, 2019.
- [12] W. Ding, Z. Yan, and R. H. Deng, "Privacy-Preserving Data Processing with Flexible Access Control," *IEEE Trans. Dependable Secur. Comput.*, vol. 17, no. 2, pp. 363–376,

2020, doi: 10.1109/TDSC.2017.2786247.

- [13] S. V. S. Prasad and K. Kumanan, "Homomorphic Encryption Using Enhanced BGV Encryption Scheme For Cloud Security," *Int. J. Eng. Comput. Sci.*, vol. 7, no. 03, pp. 23785–23789, 2018, doi: 10.18535/ijecs/v7i3.22.
- [14] L. Chen, Z. Tong, W. Liu, and C. Gao, "Non-interactive exponential homomorphic encryption algorithm," *Proc. 2012 Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov. CyberC 2012*, pp. 224–227, 2012, doi: 10.1109/CyberC.2012.44.
- [15] U. Shwetha, R. M. H, and M. June, "Securing the data in cloud using Algebra Homomorphic Encryption scheme based on updated Elgamal ( AHEE )," vol. 6, no. 3, pp. 287–292, 2017.
- [16] H. M. Al-Mashadi and A. A. Khalaf, "Hybrid homomorphic cryptosystem for secure transfer of color image on public cloud," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 19, pp. 6474–6486, 2018.
- [17] G. Vnkv, S. Rao, and G. Uma, "An Efficient Secure Message Transmission in," *GJCST-E Network, Web Secur.*, vol. 13, no. 9, 2013.
- [18] A. El-yahyaoui and M. D. Elkettani, "Fully homomorphic encryption : state of art and comparison," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 4, pp. 159–168, 2016, doi: 10.6084/M9.FIGSHARE.3362338.
- [19] R. A. A. C.Saravanabhavan, K.Anguraju, M.Kannan, P.Preethi, "Ensuring Efficient Data Storage Using Fully Mature Homomorphic Encryption Technique in the Cloud Environment," *Int. J. Recent Technol. Eng.*, vol. 8, no. 2, pp. 4820–4832, 2019, doi: 10.35940/ijrte.B2472.078219.
- [20] Y. Wang, Y. Sun, Z. Lin, and J. Min, "Container-Based Performance Isolation for Multi-Tenant SaaS Applications in Micro-Service Architecture," *J. Phys. Conf. Ser.*, vol. 1486, no. 5, 2020, doi: 10.1088/1742-6596/1486/5/052032.
- [21] J. K. R. Sastry and M. T. Basu, "Securing multi-tenancy systems through multi DB instances and multiple databases on different physical servers," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 2, p. 1385, 2019, doi: 10.11591/ijece.v9i2.pp1385-1392.
- [22] A. Jumagaliyev and Y. Elkhatib, "CadaML: A modeling language for multi-tenant cloud application data architectures," *IEEE Int. Conf. Cloud Comput. CLOUD*, vol. 2019-July, no. October, pp. 430–434, 2019, doi: 10.1109/CLOUD.2019.00075.

- [23] S. Kanade and R. Manza, "A Comprehensive Study on Multi Tenancy in SAAS Applications," *Int. J. Comput. Appl.*, vol. 181, no. 44, pp. 25–27, 2019, doi: 10.5120/ijca2019918531.
- [24] V. H. S. C. Pinto, H. J. F. Luz, R. R. Oliveira, P. S. L. Souza, and S. R. S. Souza, "A systematic mapping study on the multi-tenant architecture of SaaS systems," *Proc. Int. Conf. Softw. Eng. Knowl. Eng. SEKE*, vol. 2016-Janua, no. August, pp. 396–401, 2016, doi: 10.18293/SEKE2016-068.
- [25] M. T. Scholar and N. Kumar, "A Review on Analysis of Data Search Scheme for Secure Information Retrieval in Cloud Computing," vol. 5, no. 2, pp. 529–532, 2019.
- [26] K. Ramkumar, "Preserving security using crisscross AES and FCFS scheduling in cloud computing Preserving security using crisscross AES and FCFS scheduling in cloud computing," no. January, 2019, doi: 10.1504/IJAIP.2019.10017743.
- [27] R. R. Chowdhury, "Security in Cloud Computing," vol. 96, no. 15, pp. 24–30, 2014.
- [28] S. R. Nidhi Dahiya, "IMPLEMENTING MULTILEVEL DATA SECURITY IN CLOUD COMPUTING," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 8, 2017, doi: 10.1049/PBSE007E.
- [29] M. A. Nadeem, "Cloud Computing : Security Issues and Challenges," no. December, 2016, doi: 10.21174/jowc.v1i1.73.
- [30] S. S. Rohit BHADAURIA, Rituparna CHAKI, Nabendu CHAKI, "A SURVEY ON SECURITY ISSUES IN," 2014.
- [31] M. Nazir, "Cloud Computing : Overview & Current Research Challenges," vol. 8, no. 1, pp. 14–22, 2012.
- [32] M. Computing, "Study of Security Issues in Cloud Computing," vol. 4, no. 6, pp. 230–234, 2015.
- [33] S. D. K and K. Akilandeswari, "An Efficient Virtual Machine Intrusion Detection System on Cloud Computing," vol. 2, no. 1, pp. 143–146, 2019.
- [34] S. Kumar, S. Pal, A. Kumar, and J. Ali, "Virtualization , The Great Thing and Issues in Cloud Computing," pp. 338–341, 2013.
- [35] Q. L. Guanyu Su, Fang Wang, "Research on SQL injection Vulnerability Attack model," pp. 217–221, 2018.



- [36] B. P. Rimal, G. S. Member, M. Maier, and S. Member, "Workflow Scheduling in Multi-Tenant Cloud Computing Environments," vol. 9219, no. c, pp. 1–14, 2016, doi: 10.1109/TPDS.2016.2556668.
- [37] S. kalaivan. R.priya and Department, "SECURITY ISSUES IN CLOUD COMPUTING D.Sakthipriya," *J. Anal. Comput.*, pp. 1–5, 2018, [Online]. Available: <https://www.ijaonline.com/wp-content/uploads/2020/03/A-Review-Paper-on-Cloud-Computing-Models-1.pdf>
- [38] M. H. Sqalli, "EDoS-Shield - A Two-Steps Mitigation Technique against EDoS Attacks in Cloud Computing," 2011, doi: 10.1109/UCC.2011.17.
- [39] M. Alkharji, M. Al Hammoshi, C. Hu, and H. Liu, "Genetic Algorithm based key Generation for Fully Homomorphic Encryption," pp. 1–11, 2017.
- [40] M. J. Arshad and M. Umair, "Improving Cloud Data Encryption Customized Genetic Algorithm Using," *I.J. Intell. Syst. Appl.*, no. December, pp. 46–63, 2020, doi: 10.5815/ijisa.2020.06.04.
- [41] K. Kalaiselvi, S. Gopika, and M. Jacob, "Optimized Symmetric Keys Generated using Genetic Algorithm for Fully Homomorphic Encryption System," vol. 14, no. 06, pp. 339–343, 2021.
- [42] U. Awasthi, "Token Based Authentication Using Hash Key , Session And Javamail Api," pp. 12377–12384, 2017, doi: 10.15680/IJIRCCE.2017.
- [43] P. Gauravaram, "Security analysis of salt || password hashes," pp. 25–30, 2013, doi: 10.1109/ACSAT.2012.49.
- [44] B. Rakesh, K. Lalitha, M. Ismail, and H. P. Sultana, "DISTRIBUTED SCHEME TO AUTHENTICATE DATA STORAGE SECURITY IN CLOUD COMPUTING," vol. 9, no. 6, pp. 59–66, 2017, doi: 10.5121/ijcsit.2017.9606.
- [45] M. Rivis and S. Ying, "Achieving Regulatory Compliance for Data Protection in the Cloud," *Int. J. Adv. Comput. Sci. Appl.*, vol. 4, no. 12, pp. 162–167, 2013, doi: 10.14569/ijacsa.2013.041224.
- [46] A. Alharbi, H. Zamzami, and E. Samkri, "Survey on homomorphic encryption and address of new trend," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 7, pp. 618–626, 2020, doi: 10.14569/IJACSA.2020.0110774.

- [47] H. E. D. Kang, D. Kim, S. Kim, D. D. Kim, J. H. Cheon, and B. W. Anthony, "Homomorphic Encryption as a secure PHM outsourcing solution for small and medium manufacturing enterprise," *J. Manuf. Syst.*, vol. 61, no. June, pp. 856–865, 2021, doi: 10.1016/j.jmsy.2021.06.001.
- [48] Z. Min, G. Yang, A. K. Sangaiah, S. Bai, and G. Liu, "A privacy protection-oriented parallel fully homomorphic encryption algorithm in cyber physical systems," *Eurasip J. Wirel. Commun. Netw.*, vol. 2019, no. 1, 2019, doi: 10.1186/s13638-018-1317-9.
- [49] A. Costache, K. Laine, and R. Player, "Homomorphic noise growth in practice: comparing BGV and FV," pp. 1–36, [Online]. Available: <https://eprint.iacr.org/2019/493.pdf>
- [50] N. Sammeta and L. Parthiban, "Medical data analytics for secure multi-party-primarily based cloud computing utilizing homomorphic encryption," *J. Sci. Ind. Res. (India)*, vol. 80, no. 8, pp. 692–698, 2021.
- [51] L. Sadeghikhorami and A. A. Safavi, "Secure distributed Kalman filter using partially homomorphic encryption," *J. Franklin Inst.*, vol. 358, no. 5, pp. 2801–2825, 2021, doi: 10.1016/j.jfranklin.2020.08.048.
- [52] R. Awadallah, "Verifiable Homomorphic Encrypted Computations for Cloud Computing," no. November, 2021, doi: 10.14569/IJACSA.2021.0121089.
- [53] D. Bhatia and M. Dave, "Partial and Fully Homomorphic Encryption Schemes for Privacy Preserving," pp. 457–462, 2019.
- [54] S. Wang, X. Wang, and Y. Zhang, "A Secure Cloud Storage Framework with Access Control Based on Blockchain," *IEEE Access*, vol. 7, pp. 112713–112725, 2019, doi: 10.1109/ACCESS.2019.2929205.
- [55] A. S. Yahaya, N. Javaid, R. Khalid, M. Imran, and N. Naseer, "A Blockchain based Privacy-Preserving System for Electric Vehicles through Local Communication," *IEEE Int. Conf. Commun.*, vol. 2020-June, 2020, doi: 10.1109/ICC40277.2020.9149129.
- [56] C. H. V. N. U. Bharathi Murthy, M. L. Shri, S. Kadry, and S. Lim, "Blockchain based cloud computing: Architecture and research challenges," *IEEE Access*, vol. 8, pp. 205190–205205, 2020, doi: 10.1109/ACCESS.2020.3036812.
- [57] I. Sukhodolskiy and S. Zapechnikov, "A blockchain-based access control system for cloud storage," *Proc. 2018 IEEE Conf. Russ. Young Res. Electr. Electron. Eng.*

- EIconRus* 2018, vol. 2018-Janua, pp. 1575–1578, 2018, doi: 10.1109/EIconRus.2018.8317400.
- [58] V. Reantongcome, V. Visoottiviseth, W. Sawangphol, A. Khurat, S. Kashihara, and D. Fall, “Securing and Trustworthy Blockchain-based Multi-Tenant Cloud Computing,” *ISCAIE 2020 - IEEE 10th Symp. Comput. Appl. Ind. Electron.*, pp. 256–261, 2020, doi: 10.1109/ISCAIE47305.2020.9108796.
- [59] J. H. Park and J. H. Park, “Blockchain security in cloud computing: Use cases, challenges, and solutions,” *Symmetry (Basel)*, vol. 9, no. 8, pp. 1–13, 2017, doi: 10.3390/sym9080164.
- [60] I. Weber, Q. Lu, A. B. Tran, A. Deshmukh, M. Gorski, and M. Strazds, “A platform architecture for multi-tenant blockchain-based systems,” *Proc. - 2019 IEEE Int. Conf. Softw. Archit. ICSA 2019*, no. January 2021, pp. 101–110, 2019, doi: 10.1109/ICSA.2019.00019.
- [61] M. Shah, M. Shaikh, V. Mishra, and G. Tuscano, “Decentralized Cloud Storage Using Blockchain,” *Proc. 4th Int. Conf. Trends Electron. Informatics, ICOEI 2020*, no. Icoei, pp. 384–389, 2020, doi: 10.1109/ICOEI48184.2020.9143004.
- [62] W. Qu, L. Wu, W. Wang, Z. Liu, and H. Wang, “A electronic voting protocol based on blockchain and homomorphic signcryption,” *Concurr. Comput.*, no. November 2019, pp. 1–17, 2020, doi: 10.1002/cpe.5817.
- [63] S. Yaji, K. Bangera, and B. Neelima, “Privacy preserving in blockchain based on partial homomorphic encryption system for ai applications,” *Proc. - 25th IEEE Int. Conf. High Perform. Comput. Work. HiPCW 2018*, pp. 81–85, 2019, doi: 10.1109/HiPCW.2018.8634280.
- [64] M. M. Kumar, M. V. N. K. Prasad, and U. S. N. Raju, “BMIAE: Blockchain-based multi-instance Iris authentication using additive ElGamal homomorphic encryption,” *IET Biometrics*, vol. 9, no. 4, pp. 165–177, 2020, doi: 10.1049/iet-bmt.2019.0169.
- [65] B. Huang *et al.*, “BPS: A reliable and efficient pub/sub communication model with blockchain-enhanced paradigm in multi-tenant edge cloud,” *J. Parallel Distrib. Comput.*, vol. 143, pp. 167–178, 2020, doi: 10.1016/j.jpdc.2020.05.005.
- [66] L. Ismail, H. Materwala, and A. Hennebelle, “A scoping review of integrated blockchain-cloud (Bcc) architecture for healthcare: Applications, challenges and

- solutions,” *Sensors*, vol. 21, no. 11, 2021, doi: 10.3390/s21113753.
- [67] G. Xie, Y. Liu, G. Xin, and Q. Yang, “Blockchain-Based Cloud Data Integrity Verification Scheme with High Efficiency,” *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/9921209.
- [68] C. Yang, L. Tan, N. Shi, B. Xu, Y. Cao, and K. Yu, “AuthPrivacyChain: A Blockchain-Based Access Control Framework with Privacy Protection in Cloud,” *IEEE Access*, vol. 8, pp. 70604–70615, 2020, doi: 10.1109/ACCESS.2020.2985762.
- [69] W. Li, J. Wu, J. Cao, N. Chen, Q. Zhang, and R. Buyya, *Blockchain-based trust management in cloud computing systems: a taxonomy, review and future directions*, vol. 10, no. 1. *Journal of Cloud Computing*, 2021. doi: 10.1186/s13677-021-00247-5.
- [70] M. Uddin, A. Khalique, A. K. Jumani, S. S. Ullah, and S. Hussain, “Next-generation blockchain-enabled virtualized cloud security solutions: Review and open challenges,” *Electron.*, vol. 10, no. 20, pp. 1–23, 2021, doi: 10.3390/electronics10202493.
- [71] S. Y. Ketki R. Ingole, “Blockchain Technology in Cloud Computing : A Systematic Review,” *Int. Res. J. Eng. Technol.*, vol. 2018, pp. 1–43, 2018.
- [72] A. Toumanari, “Efficient Ciphertext-Policy Attribute-Based Encryption Constructions with Outsourced Encryption and Decryption,” vol. 2021, 2021.
- [73] P. J. Sun, “Security and privacy protection in cloud computing: Discussions and challenges,” *J. Netw. Comput. Appl.*, vol. 160, no. April, p. 102642, 2020, doi: 10.1016/j.jnca.2020.102642.
- [74] S. Dubey and P. K. Rai, “A Review of Cloud Service Security with Various Access Control Methods,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 10, no. 3, pp. 39–45, 2021, doi: 10.47760/ijcsmc.2021.v10i03.005.
- [75] A. Singh, U. Chandra, S. Kumar, and K. Chatterjee, “A Secure Access Control Model for E-health Cloud,” *IEEE Reg. 10 Annu. Int. Conf. Proceedings/TENCON*, vol. 2019-Octob, pp. 2329–2334, 2019, doi: 10.1109/TENCON.2019.8929433.
- [76] A. Chhabra, “Access Control in Multi Tenant and Diverse Cloud”.
- [77] K. Sethi, P. Bera, A. Chopra, and B. K. Tripathy, “Integration of Role Based Access Control with Homomorphic Cryptosystem for Secure and Controlled Access of Data in Cloud,” *ACM Int. Conf. Proceeding Ser.*, pp. 194–199, 2017, doi:

10.1145/3136825.3136902.

- [78] Kamlesh Kumar Hingwe and S. Mary Saira Bhanu, “Hierarchical Role-Based Access Control with Homomorphic Encryption for Database as a Service Kamlesh,” *Adv. Intell. Syst. Comput.*, vol. 409, pp. v–vi, 2016, doi: 10.1007/978-981-10-0135-2.
- [79] Z. Yu, C. Z. Gao, Z. Jing, B. B. Gupta, and Q. Cai, “A Practical Public Key Encryption Scheme Based on Learning Parity with Noise,” *IEEE Access*, vol. 6, no. c, pp. 31918–31923, 2018, doi: 10.1109/ACCESS.2018.2840119.
- [80] Z. R. Saeed, “Improved Cloud Storage Security of Using Three Layers Cryptography Algorithms,” vol. 16, no. 10, pp. 34–39, 2018.
- [81] S. Zaineldeen and A. Ate, “Improve the security of transfer data file on the cloud by executing hybrid encryption algorithms,” *Indones. J. Electr. Eng. Comput. Sci.*, vol. 20, no. 1, pp. 521–527, 2020, doi: 10.11591/ijeecs.v20.i1.pp521-527.
- [82] A. Kardi and R. Zagrouba, “Hybrid Cryptography Algorithm for Secure Data Communication in WSNs: DECRSA,” no. September, pp. 643–657, 2021, doi: 10.1007/978-981-33-6981-8\_51.
- [83] R. Abid *et al.*, “An optimised homomorphic CRT-RSA algorithm for secure and efficient communication,” *Pers. Ubiquitous Comput.*, pp. 1–10, 2021, doi: 10.1007/s00779-021-01607-3.
- [84] K. A. Kumari, A. Sharma, C. Chakraborty, and M. Ananyaa, “Preserving Health Care Data Security and Privacy Using Carmichael’s Theorem-Based Homomorphic Encryption and Modified Enhanced Homomorphic Encryption Schemes in Edge Computing Systems,” *Big Data*, vol. 10, no. 1, pp. 1–17, 2022, doi: 10.1089/big.2021.0012.
- [85] N. Helil and K. Rahman, “CP-ABE access control scheme for sensitive data set constraint with hidden access policy and constraint policy,” *Secur. Commun. Networks*, vol. 2017, 2017, doi: 10.1155/2017/2713595.
- [86] P. Dhiman and S. K. Henge, “Comparative Analysis of Cloud Security Complexities and Past Proposed Non-Homomorphic and Homomorphic Encryption Methodologies with Limitations,” *Proc. 4TH Int. Conf. Inf. Commun. Technol. Compet. Strateg. (ICTCS 2019)*, no. ISBN: 978-1-003-05209-8.
- [87] M. Alkharji, M. Al Hammoshi, C. Hu, and H. Liu, “Genetic Algorithm based key

Generation for Fully Homomorphic Encryption,” no. March, 2018.

- [88] B. Waters, “Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 6571 LNCS, no. subaward 641, pp. 53–70, 2011, doi: 10.1007/978-3-642-19379-8\_4.
- [89] P. Dhiman and S. K. Henge, “Comparative Analysis and Scrutiny of Key Authentication Techniques in Fully Homomorphic Schemes,” *Int. J. Adv. Sci. Technol.*, vol. 29, no. 7, pp. 12413–12421, 2020.
- [90] P. Zhang, J. Xu, H. Muazu, and W. Mao, “Access Control Research on Data Security in Cloud Computing,” *Proc. ICCT20 I 5 Access*, pp. 873–877.
- [91] P. Dhiman *et al.*, “Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption,” *Electronics*, vol. 11, no. 13, p. 1942, 2022, doi: 10.3390/electronics11131942.
- [92] Pooja Dhiman and Dr. Santosh Kumar Henge, “Analysis of Blockchain Secure Models and Approaches based on various services in Multi-Tenant Environment,” *Recent Innov. Comput. Lect. Notes Electr. Eng.*, vol. 855, no. 2, 2021, [Online]. Available: [https://doi.org/10.1007/978-981-16-8892-8\\_42](https://doi.org/10.1007/978-981-16-8892-8_42)

## Annexure II

### List of Publications

S No.	Title of paper with author names	Name of journal/conference	Published date	Issn no/ vol no, issue no	Indexing in Scopus/ Web of Science/UGC-CARE list
1.	Cloud Computing - Overview and its Challenges	RESEARCH REVIEW International Journal of Multidisciplinary	March 2019	Vol. 4, Issue 3 <a href="https://doi.org/10.5281/zenodo.2605480">https://doi.org/10.5281/zenodo.2605480</a>	UGC
2.	Comparative Analysis of Cloud Security Complexities and Past Proposed Non-Homomorphic and Homomorphic Encryption Methodologies with Limitations	Fourth International Conference on Information and Communication Technology for Competitive Strategies (ICTCS 2019)	May 2020	<a href="https://doi.org/10.1201/9781003052098">https://doi.org/10.1201/9781003052098</a>	Scopus and Web of Science
3.	Comparative Analysis and Scrutiny of Key Authentication Techniques in Fully Homomorphic Schemes	International Journal of Advanced Science and Technology	June 2020	Vol. 29, No. 7	Scopus

4.	Analysis of Blockchain Secure Models and Approaches based on various services in Multi-Tenant Environment	International Conference on Recent Innovations in Computing (ICRIC-2021)	April 2022	Volume 855, <a href="https://doi.org/10.1007/978-981-16-8892-8_42">https://doi.org/10.1007/978-981-16-8892-8_42</a>	Scopus
5.	Secure Token-Key Implications in Enterprise Multi-Tenancy Environment using BGV-EHC Hybrid Homomorphic Encryption	MDPI Electronics Journal	July 2022	<i>Electronics</i> 2022, 11(13); <a href="https://doi.org/10.3390/electronics11131942">https://doi.org/10.3390/electronics11131942</a>	SCI 2.397 Impact Factor
6.	Blockchain Merkle-Tree Ethereum Approach in Enterprise Multitenant Cloud Environment	Computers, Materials & Continua	Oct 2022	Vol. 74, no. 2 DOI: 10.32604/cmc.2023.030558	SCI- 3.772 Impact Factor
7.	Integrating of Rule based Secure Parameters for Analyzing Third-Party Applications and Libraries in Cross	International conference on Materials for Merging Technologies-	Publication in process	Accepted	Scopus



	Platform Development	2021 (ICMET 2021)			
8.	Enterprise Level Centric Secure System Administration for Analysis, Detection and Prevention of Vulnerabilities, Insider Attacks in Multi-Tenants Distribution Environment	International conference on Materials for Merging Technologies- 2021 (ICMET 2021)	Publication in process	Accepted	Scopus
9.	Hybrid EHC-BGV Homomorphic based Automated Key Filter Bit-Mapping Approach for Secure Data Access Control in Distributed Enterprise Multitenant	Journal of Healthcare Engineering	Publication under process	Accepted	SCI- 2.682 Impact Factor

# Annexure III

## Publications Reprints

 <b>RESEARCH REVIEW JOURNALS</b>	Volume-04 Issue-03 March-2019	ISSN: 2455-3085 (Online) RESEARCH REVIEW International Journal of Multidisciplinary www.rjournals.com [UGC Listed Journal]
<b>Cloud Computing - Overview and its Challenges</b>		
Dhiman Pooja <i>Research Scholar, Lovely Professional University, Jalandhar (India)</i>		
<b>ARTICLE DETAILS</b>	<b>ABSTRACT</b>	
<b>Article History</b> Published Online: 13 March 2019	Cloud Computing is a standout amongst the most mainstream developing innovations in this day and age. It holds the possibility to dispense with the prerequisites for setting up of staggering expense computing foundation for IT-based arrangements. Regardless of the potential additions accomplished from the cloud computing, the associations are moderate in tolerating it because of security issues and difficulties related with it. Handing over imperative information to another organization is troubling. This audit paper gives a general review of cloud computing, deployment model of cloud computing and addresses the issues and difficulties that can emerge amid the deployment of cloud services.	
<b>Keywords</b> Cloud computing, Classification, Service delivery models, Challenges		
<b>Corresponding Author</b> Email: poojadhiman25[at]yahoo.co.in		
<b>1. Introduction</b>		
Cloud Computing is the mix of an innovation, platform that gives hosting and capacity administration on the Internet. In such a domain clients need not possess the framework for different computing services. Truth be told, they can be gotten to from any PC in any piece of the world.	Framework could be overseen by an outsider cloud specialist organization or oversaw inside.	
It is the utilization of different services, for example, programming improvement platforms, servers, stockpiling and programming, over the web, regularly alluded to as the "cloud.". A cloud speaks to a network we don't think a lot about, maybe a network we don't claim, or a network that gives availability in its own particular manner. We consider it the cloud in light of the fact that everything is put away remotely and conveyed by means of online associations. There isn't one single area where this data is put away; it's simply gotten to by clients associated with the web. Normally cloud computing services are conveyed by an outsider supplier who claims the foundation. Organizations use cloud computing services since this technique is less expensive than purchasing costly computing equipment. Cloud computing offers an imaginative plan of action for associations to embrace IT services without forthright venture.	<ul style="list-style-type: none"><li>• <b>Public Cloud</b> – Public cloud suppliers pool their assets to serve multiple clients on shared equipment the supplier oversees themselves. Suppliers will assign assets, arrangement remaining tasks at hand, and design multi-occupant conditions. Public cloud clients have a restricted capacity to oversee server-side security or guarantee consistence. They likewise lose the capacity to redo equipment to enhance execution and network accessibility.</li><li>• <b>Hybrid Cloud</b> – Hybrid clouds join both public and private cloud services. The cloud foundation is made out of at least two clouds, for example, a private cloud and a network or public cloud, that stay exceptional elements however are bound together by institutionalized or exclusive innovation.</li></ul>	
The cloud computing is on-request administration and it give computing capacities as required naturally. Many cloud computing progressions are firmly identified with virtualization. The capacity to pay on interest and scale rapidly is generally an aftereffect of cloud computing sellers having the capacity to pool assets that might be partitioned among numerous customers. Security is one of the serious issues which hamper the development of cloud.	<b>3. Cloud Computing Service Delivery Models</b>	
<b>2. Classification of Clouds</b>	A web server normally has three levels to it: The physical foundation, the operating system platform, and the web application software being run. A cloud container may contain one, two or all of these layers. The cloud service model incorporates SaaS (Software as a Service), PaaS (Platform as a Service) and IaaS (Infrastructure as a Service).	
Clouds can generally be classified as three major types-private, public and hybrid. A Cloud condition can involve either a single Cloud or multiple Clouds. Customers regularly pick a kind of cloud dependent on their capacity to oversee cloud frameworks and their requests for security.	Picking an appropriate cloud supplier is frequently a matter of choosing which layers you wish to control yourself, and which ought to be the duty of the hosting supplier.	
<ul style="list-style-type: none"><li>• <b>Private Cloud</b> – The private cloud comprises of devoted assets and is worked by a single association.</li></ul>	<ul style="list-style-type: none"><li>• <b>IaaS looks like leasing a car.</b> IaaS resembles leasing a car. When you lease a car, you pick the car you want and drive it wherever you wish, however the car isn't yours. Want an upgrade? Simply lease an alternate car!</li><li>• <b>PaaS looks like taking a taxi.</b> You don't drive a taxi yourself, yet essentially advise the driver where you have to proceed to relax in the back seat.</li></ul>	
RRIJM 2015, All Rights Reserved	499   Page	

# COMPARATIVE ANALYSIS OF CLOUD SECURITY COMPLEXITIES AND PAST PROPOSED NON-HOMOMORPHIC AND HOMOMORPHIC ENCRYPTION METHODOLOGIES WITH LIMITATIONS

Pooja Dhiman<sup>1</sup> and Dr. Santosh Kumar Henge<sup>2</sup>

*Lovely Professional University, Phugwara, Punjab, India*

*<sup>1</sup>poojadhiman25@yahoo.co.in, <sup>2</sup>hingesanthosh@gmail.com, <sup>2</sup>santosh.24372@lpu.co.in*

**ABSTRACT:** The cloud computing is playing a major role in the evaluation of technical life of mankind with the successful execution of various cloud services such as software, application, platform, storage, and networking resources on a pay-per-use basis. Cloud data security is always the topmost priority of Cloud Service Providers. The cyber-attacks on cloud data prove that there is a need for continuous research for finding various updated ways to protect the cloud data from unauthorized users. Fully HEA allows computations on encrypted data without releasing the decryption key to the CSP. This review paper aims at focusing on various past proposed approaches or methodologies and their limitations, lagging issues that are related to cloud security. Based on these limitations, the future scope can be decided as what exactly can be the approach to protect the cloud data from unauthorized access to build a more secured CS.

**Keywords:** Security Issues (SI), Cloud Service Providers (CSPs), Authorization-Authentication(AA), Access Control (AC), Homomorphic Encryption Algorithm (HEA), Crypto-system (CS).

## I. INTRODUCTION

NIST (National Institute of Standards and Technology) provides a standard definition for cloud computing as, "Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort" [17].

Cloud computing has completely altered the delivery of IT services to the clients. It eliminates the requirement of setting up of high cost computing infrastructure. The ability to scale up and down the services according to the requirement of the client makes it flexible and easy to use. Cloud computing provides various hosting and storage services on the internet. Cloud service providers (CSPs) provide privacy by segregating client's data and resources into tenants. A tenant is an isolated container

## Comparative Analysis and Scrutiny of Key Authentication Techniques in Fully Homomorphic Schemes

Pooja Dhiman<sup>1</sup>, Dr.Santosh Kumar Henge<sup>2</sup>

<sup>1</sup>Research Scholar, Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

<sup>1</sup>poojadhiman25@yahoo.co.in

Associate Professor, <sup>2</sup>Department of Computer Science and Engineering, Lovely Professional University, Phagwara, Punjab, India

<sup>2</sup>hingesanthosh@gmail.com, <sup>2</sup> santosh.24372@lpu.co.in

### Abstract

This research paper is focused on various past proposed approaches, methodologies, lagging issues, advantages of various Fully Homomorphic Encryption schemes. FHE supports unlimited number of operations on ciphertext. These operations can be sorting, searching, multiplicative, additive or mixed. The four FHE schemes namely, Non-interactive Exponential Homomorphic Encryption Algorithm, Brakerski-Gentry-Vaikuntanathan, Enhanced Homomorphic cryptosystem, Algebra Homomorphic encryption scheme based on updated ElGamal are analysed. It is observed that BGV and AHHE support static and dynamic data both. AHHE is indistinguishable under CPA (Chosen plaintext attack) only; it is not CCA (chosen ciphertext attack) secure. Implementation of BGV scheme is available easily still it is a little difficult to implement in real life applications due to overhead of managing storage. Researchers can refer this paper to study these schemes in detail. Based on the requirement, particular scheme or a combination of schemes can be chosen.

**Keywords:** Non-interactive Exponential Homomorphic Encryption Algorithm (NEHE), Brakerski-Gentry-Vaikuntanathan (BGV), Enhanced Homomorphic cryptosystem (EHC), Algebra Homomorphic encryption scheme based on updated ElGamal (AHHE), Cloud Service Provider (CSP), Fully Homomorphic Encryption (FHE).

### 1 Introduction:

Homomorphic encryption algorithm allows computations to be done on encrypted data directly without the need to decrypt it using decryption key, hence maintains privacy with CSPs. Traditionally, decryption key is sent along with the encrypted data to the CSP so as to do computations on the data, violating the privacy of the client. Various versions of Homomorphic algorithm are in lime light these days. Basic types of Homomorphic scheme are:

#### A. Partial Homomorphic Encryption scheme (PHE)

Only one operation that is, either addition or multiplication can be evaluated at some instant of time. Paillier, El-Gamal, RSA are some of the algorithms base on this scheme.

#### B. Somewhat Homomorphic Encryption scheme (SWHE)

Both the operations can be evaluated at the same time but with limited number of operations on the ciphertext. It is noise based scheme. Examples include: Learning with errors, Ring learning with errors.

# Analysis of Blockchain Secure Models and Approaches Based on Various Services in Multi-tenant Environment



Pooja Dhiman and Santosh Kumar Henge

**Abstract** This research paper is focused on analyzing past proposed papers on using blockchain with cloud computing to enhance the present security system of the cloud. Homomorphic encryption technique is the latest and most secure way of protecting cloud data on the server. It allows computational operations on encrypted data as there is no need to decrypt the data for processing it. Still, we cannot guarantee privacy and data integrity. Here comes the concept of blockchain, it is a decentralized approach as compared to the cloud's centralized approach. Decentralization means data on the blockchain is stored at different locations or servers which is owned by different organizations. Hence, it is not dependent on any single third party for its execution or processing. The blockchain can be combined with fully or partially homomorphic encryption schemes to build a strong security system in a multi-tenant environment.

**Keywords** Blockchain (BC) · Cloud computing (CC) · Multi-tenancy (MT) · Homomorphic encryption (HE) · Hyperledger (HL) · Merkle tree (MT) · Fully homomorphic encryption (FHE)

## 1 Introduction

Cloud computing is a centralized approach. It stores the data on different servers located globally. Since the data is stored on remote servers, security is always the major concern. The fully Homomorphic Encryption technique (FHE) is considered to be the most secured technique as of now. It permits the processing of encrypted data, maintaining the privacy of data. But for some cases, there may be a need to share the public key with the cloud provider to perform some operations. If the provider is not a trusted party, it can manipulate the user's data. In this case, we can use Blockchain along with the FHE schemes to build a more secured network.

---








P. Dhiman (✉) · S. K. Henge  
School of Computer Science and Engineering, Lovely Professional University,  
Phagwara, Punjab, India

S. K. Henge  
e-mail: [santosh.24372@lpu.co.in](mailto:santosh.24372@lpu.co.in)

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022  
P. K. Singh et al. (eds.), *Recent Innovations in Computing*, Lecture Notes in Electrical Engineering 855, [https://doi.org/10.1007/978-981-16-8892-8\\_42](https://doi.org/10.1007/978-981-16-8892-8_42) 563

Article

## Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption

Pooja Dhiman <sup>1</sup>, Santosh Kumar Henge <sup>1</sup>, Rajakumar Ramalingam <sup>2</sup>, Ankur Dumka <sup>3,4</sup>, Rajesh Singh <sup>5</sup>, Anita Gehlot <sup>5</sup>, Mamoon Rashid <sup>6,\*</sup>, Sultan S. Alshamrani <sup>7</sup>, Ahmed Saeed AlGhamdi <sup>8</sup>, and Abdullah Alshehri <sup>9</sup>

- <sup>1</sup> School of Computer Applications, Lovely Professional University, Phagwara 144402, India; poojadhiman25@yahoo.co.in (P.D.); hingesanthosh@gmail.com (S.K.H.)
  - <sup>2</sup> Department of Computer Science and Technology, Madanapalle Institute of Technology & Science, Madanapalle 517325, India; ramukshare@gmail.com
  - <sup>3</sup> Department of Computer Science and Engineering, Women Institute of Technology, Dehradun 248007, India; ankurdumka2@gmail.com
  - <sup>4</sup> Department of Computer Science and Engineering, Graphic Era Deemed to be University, Dehradun 248001, India
  - <sup>5</sup> Uttarakhand Institute of Technology, Uttarakhand University, Dehradun 248007, India; rajeshsingh@uttarakhanduniversity.ac.in (R.S.); anita.ri@uttarakhanduniversity.ac.in (A.G.)
  - <sup>6</sup> Department of Computer Engineering, Faculty of Science and Technology, Vishwakarma University, Pune 411048, India
  - <sup>7</sup> Department of Information Technology, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21944, Saudi Arabia; susamash@tu.edu.sa
  - <sup>8</sup> Department of Computer Engineering, College of Computer and Information Technology, Taif University, P.O. Box 11099, Taif 21994, Saudi Arabia; asjannah@tu.edu.sa
  - <sup>9</sup> Department of Information Technology, Al Baha University, P.O. Box 1988, Al Baha 65731, Saudi Arabia; aashehri@bu.edu.sa
- \* Correspondence: mamoon.rashid@vupune.ac.in; Tel.: +91-7814346505



**Citation:** Dhiman, P.; Henge, S.K.; Ramalingam, R.; Dumka, A.; Singh, R.; Gehlot, A.; Rashid, M.; Alshamrani, S.S.; AlGhamdi, A.S.; Alshehri, A. Secure Token–Key Implications in an Enterprise Multi-Tenancy Environment Using BGV–EHC Hybrid Homomorphic Encryption. *Electronics* **2022**, *11*, 1942. <https://doi.org/10.3390/electronics11131942>

**Academic Editors:** Hamed Taherdoost, Seyyed Reza Shabamini, Kamaljeet Sandhu, Basit Qureshi, Hazra Intran and Salimur Choudhury

Received: 17 May 2022  
Accepted: 19 June 2022  
Published: 21 June 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Abstract:** Authentication, authorization, and data access control are playing major roles in data security and privacy. The proposed model integrated the multi-factor authentication–authorization process with dependable and non-dependable factors and parameters based on providing security for tenants through a hybrid approach of fully homomorphic encryption methodology: the enhanced homomorphic cryptosystem (EHC) and the Brakersky–Gentry–Vaikuntanathan (BGV) scheme. This research was composed of four major elements: the fully homomorphic encryption blended schemes, EHC and BGV; secure token and key implications based on dependable and non-dependable factors; an algorithm for generating the tokens and the suitable keys, depending on the user's role; and the execution of experimental test cases by using the EHC algorithm for key and token generation, based on dependable and non-dependable parameters and time periods. The proposed approach was tested with 152 end-users by integrating six multi-tenants, five head tenants, and two enterprise levels; and achieved a 92 percent success rate. The research integrated 32-bit plain text in the proposed hybrid approach by taking into consideration the encryption time, decryption time, and key generation time of data transmission via cloud servers. The proposed blended model was efficient in preventing data from ciphertext attacks and achieved a high success rate for transmitting data between the multi-tenants, based on the user-role-user type of enterprise cloud servers.

**Keywords:** Session Initiation Time (SIT); Type of Tenant (TTenant); Type of Data (TData); Authentication Type (AT); On-demand OTP (OOTP); Hybrid Key (HK); OS Salting Key (OSSK)

## Blockchain Merkle-Tree Ethereum Approach in Enterprise Multitenant Cloud Environment

Pooja Dhiman<sup>1</sup>, Santosh Kumar Henge<sup>1</sup>, Sartaj Singh<sup>1</sup>, Avnash Kaur<sup>2</sup>, Parminder Singh<sup>2,3</sup> and Mustapha Hadabou<sup>3,\*</sup>

<sup>1</sup>School of Computer Applications, Lovely Professional University, Phagwara, 144001, India

<sup>2</sup>School of Computer Science and Engineering, Lovely Professional University, Phagwara, 144001, India

<sup>3</sup>School of Computer Science, Mohammed VI Polytechnic University, Ben Guerir, 43150, Morocco

\*Corresponding Author: Mustapha Hadabou, Email: mustapha.hadabou@um6p.ma

Received: 29 March 2022; Accepted: 12 July 2022

**Abstract:** This research paper puts emphasis on using cloud computing with Blockchain (BC) to improve the security and privacy in a cloud. The security of data is not guaranteed as there is always a risk of leakage of users' data. Blockchain can be used in a multi-tenant cloud environment (MTCE) to improve the security of data, as it is a decentralized approach. Data is saved in unaltered form. Also, Blockchain is not owned by a single organization. The encryption process can be done using a Homomorphic encryption (HE) algorithm along with hashing technique, hereby allowing computations on encrypted data without the need for decryption. This research paper is composed of four objectives: Analysis of cloud security using Blockchain technology; Exceptional scenario of Blockchain architecture in an enterprise-level MTCE; Implementation of cipher-text policy attribute-based encryption (CP-ABE) algorithm; Implementation of Merkle tree using Ethereum (MTuE) in a Multi-tenant system. Out of these four objectives, the main focus is on the implementation of CP-ABE algorithm. CP-ABE parameters are proposed for different levels of tenants. The levels include inner tenant, outer tenant, Inner-Outer-Tenant, Inner-Outer-External-Tenant, Outer-Inner-Tenant, External-Outer-Inner-Tenant and the parameters such as token, private key, public key, access tree, message, attribute set, node-level, cipher-text, salting which will help in providing better security using CP-ABE algorithm in a multi-tenant environment (MTE) where tenants can be provided with different levels of security and achieved 92 percentage of authenticity and access-control of the data.

**Keywords:** Blockchain (BC); merkle tree using ethereum (MTuE); multi-tenant environment (MTE); homomorphic encryption (HE); ciphertext policy attribute-based encryption (CP-ABE)



This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.