Mini Project report on

Modified Attribute Based Hierarchical Encryption Scheme in Cloud Computing

A Dissertation submitted to JNTU Hyderabad in partial fulfillment of the academic requirements for the award of the degree.

Bachelor of Technology

in

Computer Science and Engineering

Submitted by

K.AKSHITHA

(19H51A0513)

K.LOKESH REDDY

(19H51A0514)

K.DARSHITH RAJ

(19H51A0515)

Under the esteemed guidance of

Dr. Sarat Chandra Nayak

(Assistant Professor)

CMR

EXPLORE TO INVENT

Department of Computer Science and Engineering

CMR COLLEGE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institution under UGC & JNTUH, Approved by AICTE, Permanently Affiliated to JNTUH, Accredited by NBA.)

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD - 501401.

2019- 2023

CMR COLLEGE OF ENGINEERING & TECHNOLOGY

KANDLAKOYA, MEDCHAL ROAD, HYDERABAD – 501401

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Mini Project-1 report entitled **"Modified Attribute Based Hierarchical Encryption Scheme in Cloud Computing"** being submitted by **K. Akshitha(19H51A0513**), **K. Lokesh Reddy (19H51A0514), K. Darshith Raj (19H51A0515)**, in partial fulfillment for the award of **Bachelor of Technology in Computer Science and Engineering** is a record of Bonafide work carried out his/her under my guidance and supervision.

The results embody in this project report have not been submitted to anyother University or Institute for the award of any Degree.

Dr. Sarat Chandra Nayak   Dr. K Vijaya Kumar

Asst. Professor   Professor and HOD

Dept. Of CSE   **Dept. Of CSE**

TABLE OF CONTENTS

List of Figures

ABSTRACT

In this era where everything is being done online, there's a huge possibility of data leakage. One of the most common ways is from the cloud, as there are few methods to secure the confidential information but still there's a chance of the data being hacked. One of the methods are data encryption using attribute based and key policy based. Few more ideologies can be included in the method which helps data owner secure his data. And the new method is modified attribute based hierarchical encryption. This method can provide access control and keep the data safe which is being shared with the users in the cloud. The data stored is divided into nodes which are attributes, then we encrypt the tree and provide an access key for it. This helps the process to perform well in terms of security, efficiency and the size of the storage.

.

CHAPTER 1

INTRODUCTION

INTRODUCTION

Cloud computing collects and organizes a large amount of information technique resources to provide secure, efficient, flexible and on demand services [29]. Attracted by these advantages, more and more enterprise and individual users trend to outsource the local documents to the cloud. In general, the documents need to be encrypted before being out- sourced to protect them against leaking. If the data owner wants to share these documents with an authorized data user, they can employ any searchable encryption techniquesor privacy-preserving multi-keyword document search schemes to achieve this goal.

However, all these schemes cannot provide _ne-grained access control mechanisms to the encrypted documents. Attribute-based encryption (ABE) schemes can provide complicated systems to diversify the data users' access paths. In ABE schemes, each document is encrypted individually and a data user can decrypt a document if her attribute set matches the access structure of the document. Existing ABE schemes can be divided into Key-Policy ABE (KP-ABE) schemes and Cipher text-Policy ABE (CP-ABE) schemes. Com-pared with KP ABE schemes, CP-ABE schemes are more flexible and suitable for general applications.

In the following, we must analyze the existing ABE schemes in detail and further present the novelty and innovation of the CP-ABHE scheme proposed in this paper. For convenience, we choose the schemes in and as typical examples of KP-ABE scheme and CP-ABE scheme, respectively. Let $G0$ and$G1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of G0 and e be a bilinear map. Further, a hash function which can map an attribute string to a random element. Assume that we need to encrypt a set of documents Attribute set is the common attribute dictionary of both documents and data users. We further assume that document Fi is related with a set of attributes, denoted as $att(Fi)$.7

We encrypt F in two phases. First, each document *Fi* is encrypted by a proper symmetric encryption algorithm with a unique content key *cki*. Second, all the content keys of F are encrypted by ABE schemes. Note that, both the cipher texts of *Fi* and *cki* are provided to data users.

1.1 NEED STATEMENT

In the world where the techniques of securing the data keep improvising, we have seen the data being breached on various platforms from the cloud. The concept of Modified Attribute Based Hierarchical Encryption helps us to secure the data and encrypt it at most confidentiality.

1.2 OBJECTIVE, SCOPE & LIMITATION

1.2.1 Objective

An Attribute Based Encryption scheme was introduced by Sahai and Waters in 2005 and the goal is to provide security and access control Sahai et al 2007. Attribute-based encryption (ABE) is a public-key algorithm based one to many encryptions that allows users to encrypt and decrypt data based on user attributes.

1.2.2 Scope

Our project mainly covers these operations:
- ➢ Modified Attribute-based encryption can be used for log encryption. This primitive can also be used for broadcast encryption in order to decrease the number of keys used.
- ➢ Attribute-based encryption methods are also widely employed in vector driven search engine interfaces.

1.2.3 Limitations

- Existing files cannot be encrypted. To encrypt a file that is currently not encrypted, you must copy it into a new file whose encryption policy rules dictate that the file is to be encrypted. Note that renaming a file does not change its encryption attributes. Encryption attributes are defined at the time that the file is created.
- The following types of nodes must have network connectivity to the key server node so that they can retrieve the master encryption key (MEK), which is needed to encrypt or decrypt file data

CHAPTER 2

BACKGROUND WORK

BTECH CSE

2.1 INTRODUCTION

This section discusses findings and observations done by some research works on Modified Attribute Based Hierarchical Encryption.

2.2 LITERATURE SURVEY

From past 5-10 years hackers have been stealing the data from various sites and clouds .

- ➢ **"Yahoo"** till 2013, it's been most used.The company announced that, one billion customers were affected but the final number came in as more than 3 billion accounts.The hack started with a spear-phishing mail that is sent in early 2014 to a Yahoo employee "One click and Boom!"

- ➢ In November of 2019, an attack hit the "**Alibaba**" Chinese shopping website that impacted more than 1.1 billion pieces of user data that includes phone numbers and customer comments. The breach was severe enough to the company. China, like always don't want the public to know the full consequences of this attack.

- ➢ "**SinaWeibo**", China's largest social media platform announced in 2020 that the company has faced a situation by a hacker who stole the 172 million user data . The hacker has kept the Weibo's data up for sale of 250$. After this breach the company has lost the its usage and the users just used it at times.

Reference link: https://blog.storagecraft.com/7-infamous-cloud-security-breaches/

2.3 EXISTING SOLUTIONS

There are many existing solutions for prevention of data manipulation. A few existing solutions are:-

➢ Attribute Based Encryption

➢ Hierarchical based Encryption

➢ Key Policy Based Encryption

2.3.1 Attribute-based encryption

Attribute-based encryption is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes (e.g. the country in which they live, or the kind of subscription they have). In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.A crucial security aspect of attribute-based encryption is collusion-resistance: An adversary that holds multiple keys should only be able to access data if at least one individual key grants access
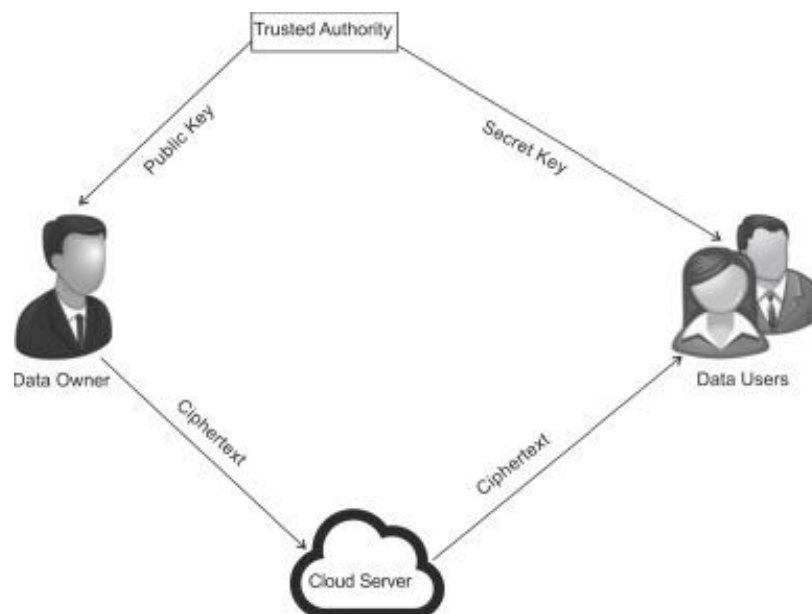
Fig : Attribute Based Encryption
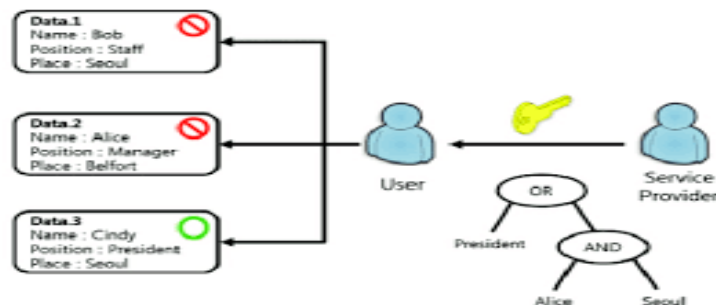
2.3.2 Hierarchical Encryption

In a hierarchical structure, a user in a security class has access to information items of another class if and only if the former class is a predecessor of latter. Based upon cryptographic techniques, several schemes have been proposed for solving the problem of access control in hierarchical structures.

Fig :Hierarchical Encryption

2.3.3 Key Policy Based Encryption

Key-policy attribute-based encryption (KP-ABE) is a type of Attribute Based Encryption, which enables senders to encrypt messages under a set of attributes and private keys are associated with access structures that specify which ciphertexts the key holder will be allowed to decrypt.

Fig : Hierarchical Encryption

7

CHAPTER 3

PROPOSED SYSTEM

3.1 INTRODUCTION

This section presents the research methodology used in the study, the research design, and the data collection process.

3.2 THEORETICAL/ CONCEPTUAL FRAMEWORK

The conceptual framework helps illustrate the research design and the relationships of the variables involved.

Fig 3.2 Conceptual Framework

3.2.1 Functional Requirements

The system designs an attribute-based document hierarchical encryption scheme named CP-ABHE which performs well in terms of computation and storage space efficiency.

3.2.3 Hardware Requirements

Processor   -   Pentium –IV

RAM   -   4 GB (min)

Hard Disk   -   20 GB

3.2.4 Software Requirements

Operating System   -   Windows XP

Coding Language   -   Python

3.3 ADVANTAGES

➢ The system is implemented based on Attribute Based Encryption scheme which gives more security on data.
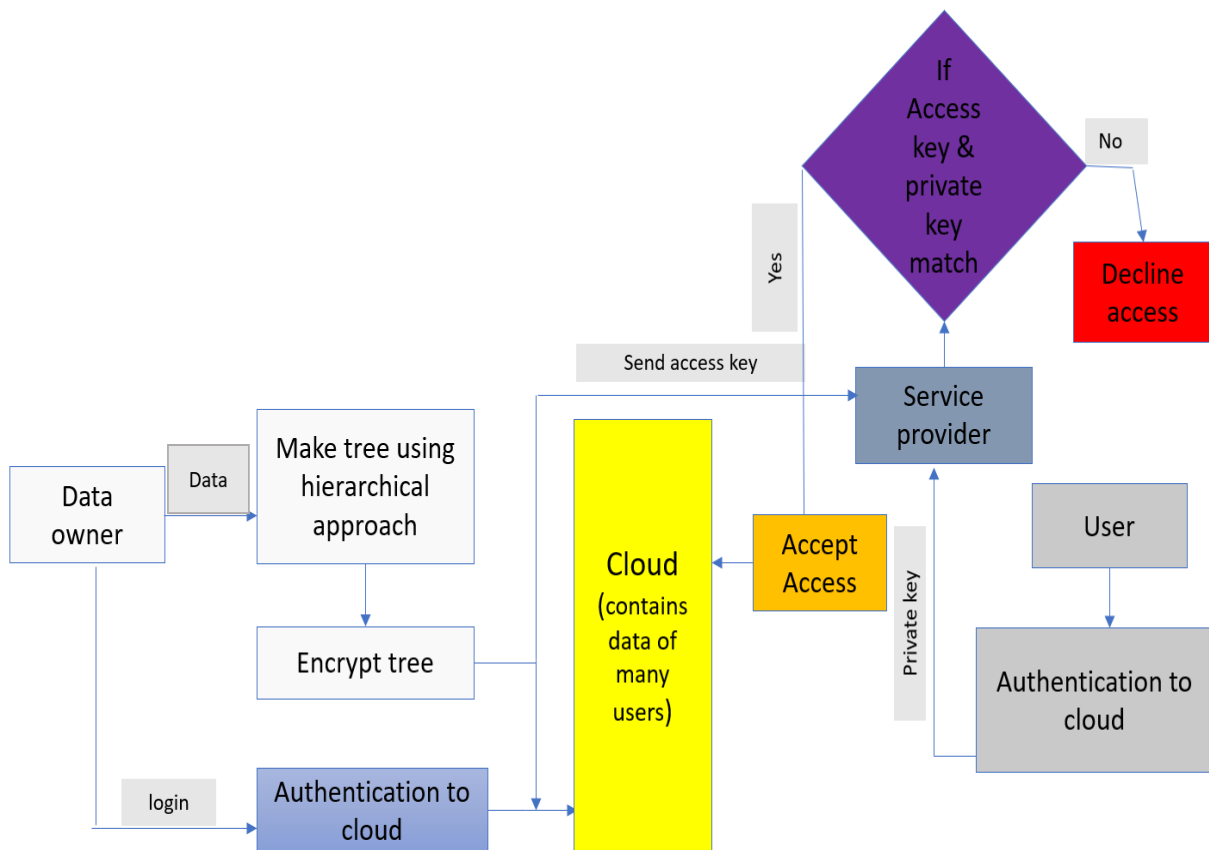
➢ The system can secure more amount of data.

CHAPTER 4

DESIGNING

4.1 PRELIMINARY DESIGN

4.1.1 UML Diagrams

Fig : UML Diagram

CHAPTER 5

RESULTS & DISCUSSION

5.1 IMPLEMENTATION

5.1.1 code:

```
#
#%%
import struct
import hashlib
from firebase import firebase
from tkinter import *
#root = Tk()
#root.geometry('400x200')class object:
#    def _init_(self, *args):
#        super(, self)._init_(*args)
#        #!/usr/bin/env python3


class MD4:


    width = 32
    mask = 0xFFFFFFFF

    # Unlike, say, SHA-1, MD4 uses little-endian. Fascinating!
    h = [0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476]
    def _init_(self, msg=None):
        if msg is None:
            msg = b""
        self.msg = msg
        # Pre-processing: Total length is a multiple of 512 bits.
        ml = len(msg) * 8
        msg += b"\x80"
```

14

```
        msg += b"\x00" * (-(len(msg) + 8) % 64)
        msg += struct.pack("<Q", ml)


        # Process the message in successive 512-bit chunks.
    self._process([msg[i : i + 64] for i in range(0, len(msg), 64)])


    def _repr_(self):
        if self.msg:
            return f"{self._class.name_}({self.msg:s})"
        return f"{self._class.name_}()"


    def _str_(self):
        return self.hexdigest()


    def _eq_(self, other):
        return self.h == other.h


    def bytes(self):
        """:return: The final hash value as a `bytes` object."""
        return struct.pack("<4L", *self.h)


    def hexbytes(self):
        """:return: The final hash value as hexbytes."""
        return self.hexdigest().encode
    def hexdigest(self):
        """:return: The final hash value as a hexstring."""
        return "".join(f"{value:02x}" for value in self.bytes())


    def _process(self, chunks):
```

15

```
        for chunk in chunks:
            X, h = list(struct.unpack("<16I", chunk)), self.h.copy()


            # Round 1.
            Xi = [3, 7, 11, 19]
            for n in range(16):
    i, j, k, l = map(lambda x: x % 4, range(-n, -n + 4))
                K, S = n, Xi[n % 4]
hn = h[i] + MD4.F(h[j], h[k], h[l]) + X[K]
                h[i] = MD4.lrot(hn& MD4.mask, S)


            # Round 2.
            Xi = [3, 5, 9, 13]
            for n in range(16):
    i, j, k, l = map(lambda x: x % 4, range(-n, -n + 4))
                K, S = n % 4 * 4 + n // 4, Xi[n % 4]
hn = h[i] + MD4.G(h[j], h[k], h[l]) + X[K] + 0x5A827999
                h[i] = MD4.lrot(hn& MD4.mask, S)


            # Round 3.
            Xi = [3, 9, 11, 15]
            Ki = [0, 8, 4, 12, 2, 10, 6, 14, 1, 9, 5, 13, 3, 11, 7, 15]
            for n in range(16):
    i, j, k, l = map(lambda x: x % 4, range(-n, -n + 4))
                K, S = Ki[n], Xi[n % 4]
hn = h[i] + MD4.H(h[j], h[k], h[l]) + X[K] + 0x6ED9EBA1
                h[i] = MD4.lrot(hn& MD4.mask, S)
        self.h = [((v + n) & MD4.mask) for v, n in zip(self.h, h)]
```

```python
    @staticmethod
    def F(x, y, z):
        return (x & y) | (~x & z)
    @staticmethod
    def G(x, y, z):
        return (x & y) | (x & z) | (y & z)
    @staticmethod
    def H(x, y, z):
        return x ^ y ^ z
    @staticmethod
    def lrot(value, n):
    lbits, rbits = (value << n) & MD4.mask, value >> (MD4.width - n)
        return lbits | rbits


def main():
    # Import is intentionally delayed.
    import sys
    if len(sys.argv) > 1:
        messages = [msg.encode() for msg in sys.argv[1:]]
        for message in messages:
    print(MD4(message).hexdigest())
    else:
        messages = [b"defrfr", b"The quick brown fox jumps over the lazy dog",
    b"BEES"]
    known_hashes = [
            "31d6cfe0d16ae931b73c59d7e0c089c0",
            "1bee69a46ba811185c194762abaeae90",
            "501af1ef4b68495b5b7e37b15b4cda68",
        ]
```

17

```
print("Testing the MD4 class.")
print()


    for message, expected in zip(messages, known_hashes):
print("Message: ", message)
print("Expected:", expected)
print("Actual:  ", MD4(message).hexdigest())
print()
'''
def btn_command():
    return None


def click_1(event):
    e1.config(state=NORMAL)
    e1.delete(0, END)


def click_2(event):
    e2.config(state=NORMAL)
    e2.delete(0, END)


def click_3(event):
    e3.config(state=NORMAL)
    e3.delete(0, END)
'''
if _name_ == "_main_":
    try:
main()
    firebase = firebase.FirebaseApplication("https://hashencoder-577c1-default-
rtdb.firebaseio.com/",None)
    '''e1 =Entry(root,width=20)
```

18

BTECH CSE

```
        #e1.insert(0, "Enter your name ")
        #e1.config(state=DISABLED)
        #e1.bind("<Button-1>",click_1)
        e1.pack()
        e2 =Entry(root,width=20)
        #e2.insert(0, "Enter your name ")
        #e2.config(state=DISABLED)
        #e2.bind("<Button-1>",click_2)
        e2.pack()
        e3 =Entry(root,width=20)
        #e3.insert(0, "Enter your name ")
        #e3.config(state=DISABLED)
        #e3.bind("<Button-1>",click_3)
        e3.pack()

        Button(root,text='Next',command=btn_command()).pack()'''
        name = input('Enter the your name : ')
        email = input('Enter your email :')
phoneNo = input('Enter your data :')
        data_transfer_name=hashlib.sha256(name.encode('utf-8')).hexdigest()
        data_transfer_email=hashlib.sha256(email.encode('utf-8')).hexdigest()
        data_transfer_phoneNum=hashlib.sha256(phoneNo.encode('utf-8')).hexdigest()

        #data_transfer_email=hashlib.sha256(email.encode('utf-8')).hexdigest()
        #data_transfer_phoneNum=hashlib.sha256(phoneNo.encode('utf-8')).hexdigest()
    print("Encoded Value of name : "+data_transfer_name)
    print("Encoded Value of E-mail : "+data_transfer_email)
    print("Encoded Data is : "+data_transfer_phoneNum)
    phoneNum = (phoneNo)
```
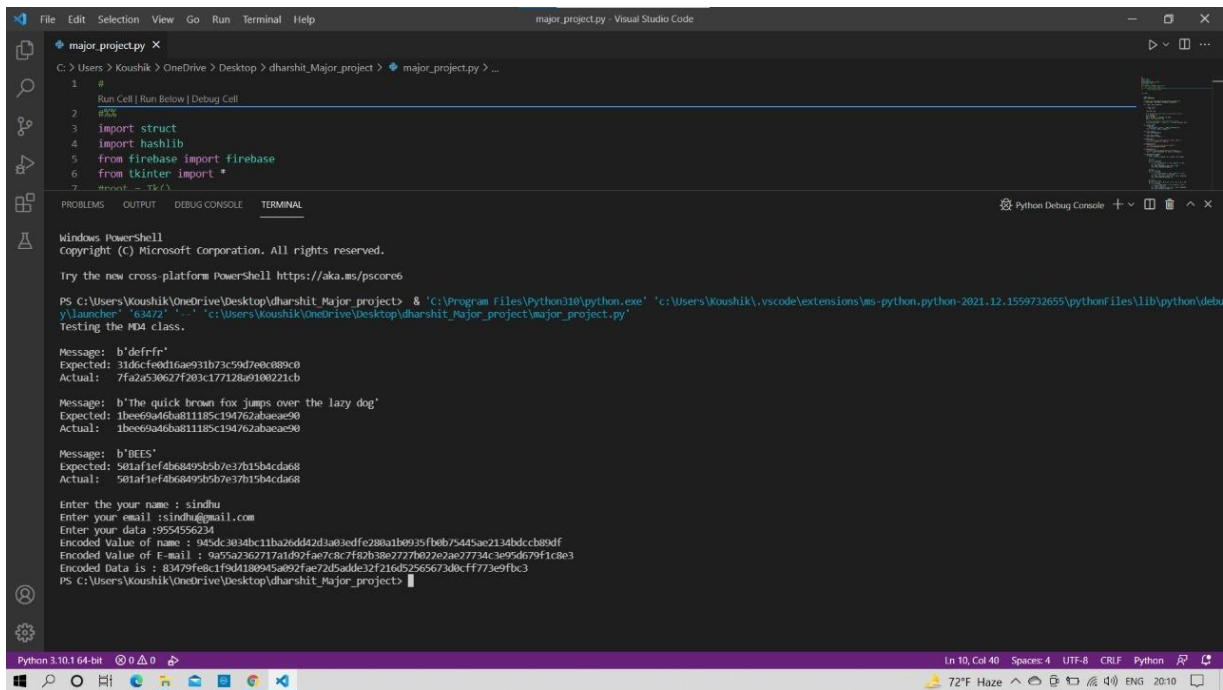
19

```
        data = {
            'Name': name,
            'Name_encoded': data_transfer_name,
            'Email':email,
            'Email_encoded':data_transfer_email,
            'Data':phoneNum,
            'Data_encoded':data_transfer_phoneNum
        }
        result = firebase.post('/hashencoder-577c1-default-rtdb/student_Details',data)
        '''phone_No = int(phoneNo)
        data = {
            'Name': name,
            'Email':email,
            'Data':phone_No
        }
        result = firebase.post('/hashencoder-577c1-default-rtdb/student_Details',data)'''
#print(btn_command())
        #print(result)
       # root.mainloop()


   except KeyboardInterrupt:
      pass
   # %%
```

Modified Attribute Based Encryption

5.2 RESULT

Fig : VS Code

Fig : Google Firebase

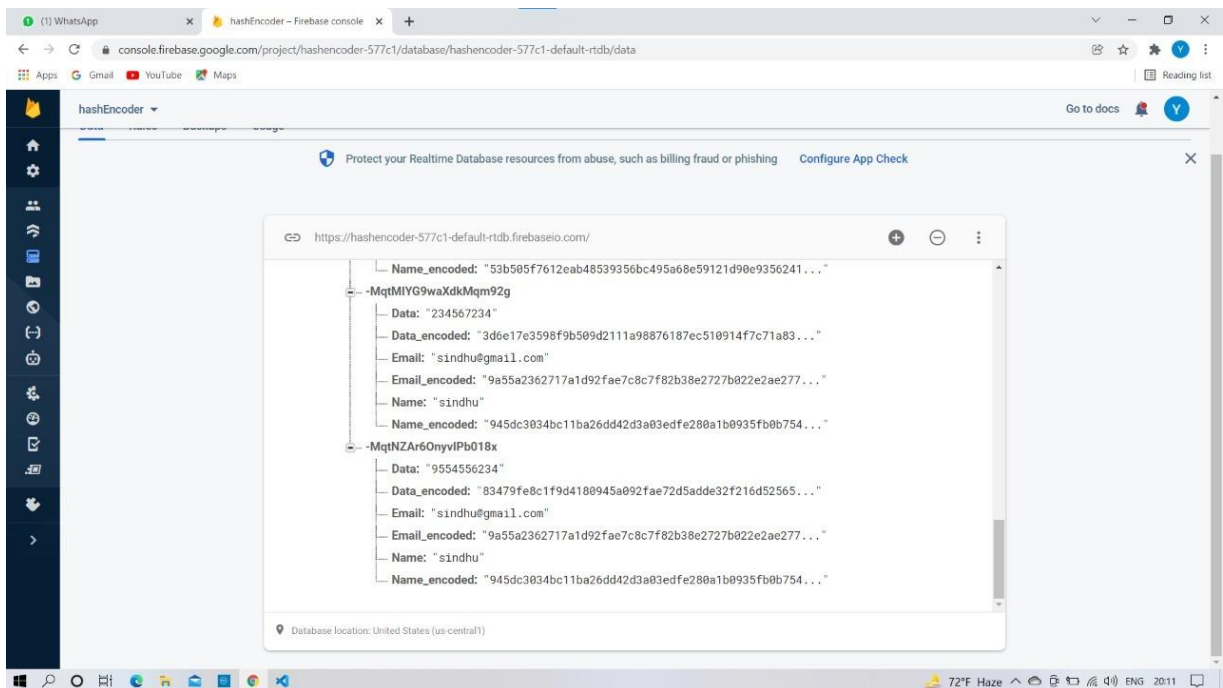CMRCET   BTECH CSE   21

CHAPTER 6
CONCLUSION &
FUTURE WORK

6.1 CONCLUSIONS

In this paper, we design a hierarchical document collection encryption scheme. An incremental algorithm to construct the integrated access trees of the documents and decrease the number of trees. Then, each integrated access tree is encrypted together and the documents in a tree can be decrypted at a time. Different to existing schemes, we construct the secret numbers for the nodes of the trees in a bottom-up manner. In this way, the sizes of cipher text and secret keys significantly decrease. At last, a thorough performance evaluation is provided including security analysis, efficiency analysis, and simulation. Results show that the pro- posed scheme outperforms KP-ABE and CP-ABE schemes in terms of encryption/decryption efficiency and storage space. Our scheme can be further improved in several aspects: First, the access policy discussed in Section III assumes that the access trees are composed of only "AND" gates. Extending the flexibility and versatility of the access policy is one of the most important research directions. Second, the documents are encrypted before outsourcing and a promising task is how to efficiently search the interested documents over the cipher texts. At last, we focus our attention on the static document collection and how to efficiently encrypt/decrypt a dynamic document collection will be also researched in the future.

6.2 FUTURE WORKS

Today, modern encryption uses 'keys' to safeguard data on our computers, mobile devices, and communication networks. Encryption converts data into digital gibberish, so it can't be used maliciously. If the message recipient has the right keys, then the data can be decrypted for processing by a computer or mobile device.

With so much of our information now stored or processed in the cloud, how can we

make sure it's safe from unauthorized access? The US National Science Foundation is funding researchers looking for answers to cyber-security challenges and exploring the future of encryption. The goal is that, one day, we will be able to ensure the security of important information wherever it may be: on our computers, mobile devices, and even in the cloud.

REFERENCES

- J. Bethencourt, A. Sahai, and B.Waters "Ciphertext-policy attribute-based

- encryption,"in*Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 321_334.

- https://www.google.com/imgres?imgurl=x-raw-

- https://www.hindawi.com/journals/mpe/2013/810969

- https://www.researchgate.net/publication/45787183_An_Asymmetric_Cryptogr aphic_Key_Assignment_Scheme_for_Access_Control_in_Tree_Structural_Hie rarchies

- https://link.springer.com/chapter/10.1007/978-3-642-36362-7_11

- https://www.agileit.com/news/data-encryption-methods-secure-cloud/

- https://www.researchgate.net/publication/343683805_Encryption_algorithm_in _cloud_computing