A Seminar on

Revocable Multi-Authority Attribute-Based Encryption with Time Based Authority

Team Details

1. K V Sai Srujan(20EG105304)
2. J Nikhil Anand(20EG105318)
3. Mahmood Ali Khan(20EG105329)

Project Supervisor
Mrs. P. Swarajya Lakshmi
Assistant Professor

Introduction

- Cloud storage is used to store massive data, so more and more individuals and organizations shift their data from local computers to cloud. However, this new model poses a serious threat to the privacy of their owners, since the data might be accessed and analyzed by the cloud server providers for illegal or monetary purposes.

- To solve this problem, people have figured out a variety of approaches. One common way is to use the traditional public key encryption technology to encrypt data, but the data owners fail to have fine-grained access to their data flexibly.

- Then many single-authority attribute-based encryption schemes have been put forward. In these schemes, it is required that only one trusted attribute authority administers the attributes and distributes the corresponding secret keys of attributes to the data consumers.This mechanism may not meet the practical requirements in cloud storage because scaling the system might be challenging, especially if there is a need for a large number of attributes or users

What it is:

Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority (RMA-ABE-TBA) is a cryptographic scheme that combines several principles to provide advanced access control and secure data sharing.Here's a breakdown of its key components.

- Attribute-Based Encryption (ABE)
- Multi-Authority
- Time-Based Authority
- Revocability
-

2. What is Needed:

For the implementation of Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority, the following components and considerations are typically required:

- Multiple Authorities
- User Attributes
- Time Constraints
- Secure Communication Channels

3. Applications:

- Cloud Computing
- Communication Systems
- Data Sharing Platforms
- Healthcare Systems
- Financial Systems
- Government and Defense

Problem Statement

Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority (RMA-ABE-TBA) is an advanced cryptographic scheme that addresses the need for secure and flexible data access control in various applications. This encryption method combines the benefits of attribute-based encryption (ABE) and revocable multi-authority to provide a robust solution for managing access to sensitive information.

Existing Method(s) Disadvantages:

While traditional attribute-based encryption and revocable multi-authority schemes offer valuable features, they come with certain limitations. In standard attribute-based encryption, scalability can be an issue as the number of attributes increases. Additionally, the revocation of access rights in multi-authority schemes may suffer from inefficiencies and complexities. Time-based authorities, on the other hand, are not always well-integrated into existing systems, leading to potential synchronization challenges.

Proposed Method

The proposed method, Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority, introduces a cutting-edge cryptographic solution designed to address the pressing challenges associated with secure access control in cloud storage systems. At its core, the cryptographic scheme utilizes a Multi-Authority Attribute-Based Encryption (MA-ABE) framework, enabling access control based on multiple authorities and attributes.What sets this method apart is the incorporation of a time-based authority component, a key innovation that introduces dynamic management of access privileges over distinct periods.The scheme's approach to revocable access control is particularly noteworthy. By integrating time-based authority, it allows for the seamless adjustment of access privileges, ensuring that security measures align with changing circumstances. This capability is vital in dynamic and collaborative environments where user roles may undergo frequent modifications.

Proposed Method

The workflow involves setting up attributes, granting user access, managing access based on time, and efficiently revoking access when needed. This provides a clear overview of how the method is adaptable and efficient in handling dynamic access control scenarios. Overall, the proposed method aims to enhance security and adaptability in cloud storage systems.

Workflow:

Attribute Setup:

- Authorities define initial attributes and associated time periods.

User Access:

- Users gain access to specific data based on their attributes and the current time period.

Time-Based Authority:

- Authorities dynamically manage access privileges over time, adapting to changing user roles.

Revocation:

- Revocation process is visually represented, showcasing the efficient removal of access privileges when needed.

Experiment Environment

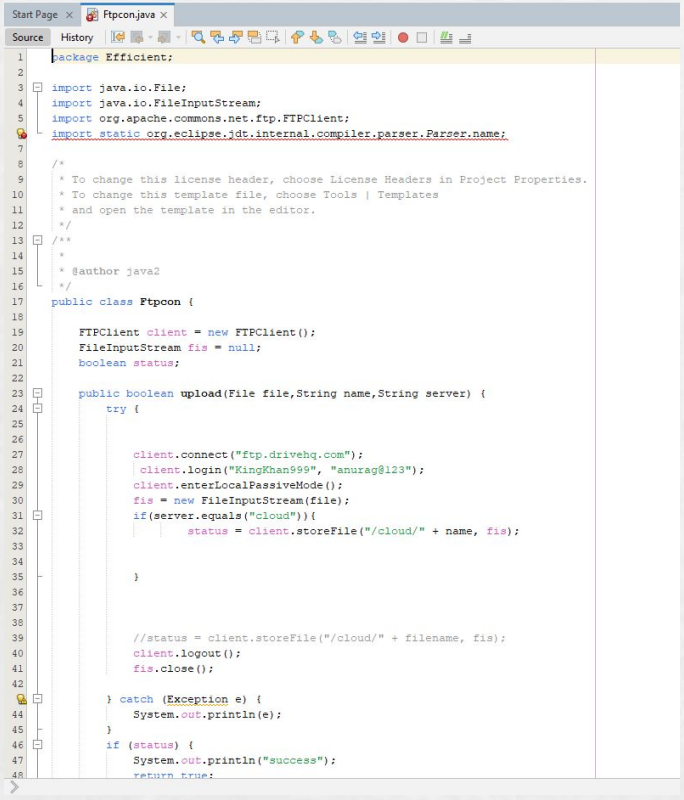
SOFTWARE REQUIREMENTS:

- Operating system : Windows XP/7/10.
- Coding Language : Java
- Tool : Netbeans
- Database : MYSQL

HARDWARE REQUIREMENTS:

- System : i5 10 Gen
- Hard Disk : 512GB.
- Ram : 16 GB.

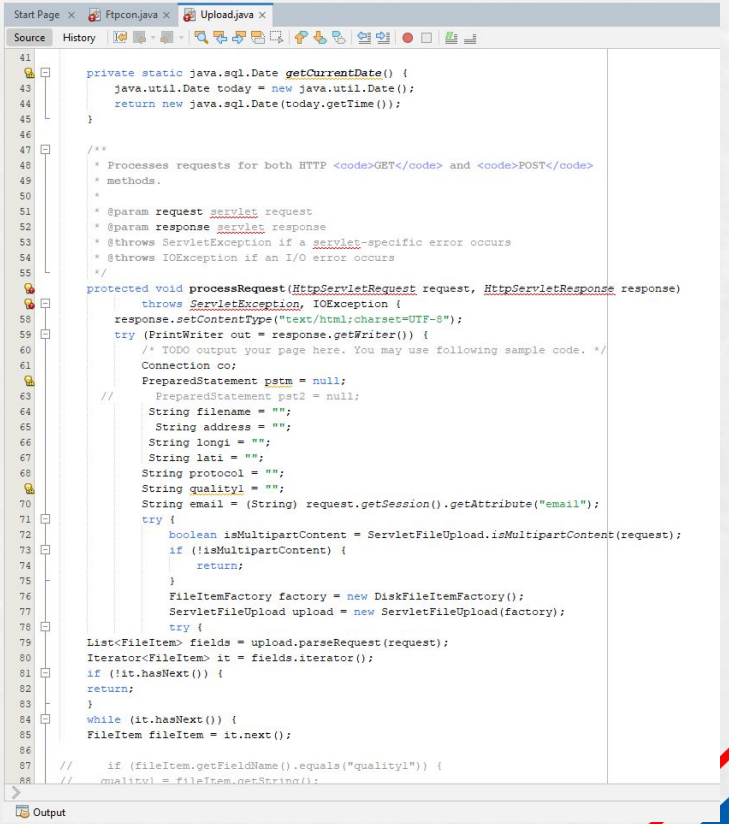
Experiment Screenshots



Start Page | Ftpcon.java

Source | History

```
package Efficient;

import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.net.ftp.FTPClient;
import static org.eclipse.jdt.internal.compiler.parser.Parser.name;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
/**
 *
 * @author java2
 */
public class Ftpcon {

    FTPClient client = new FTPClient();
    FileInputStream fis = null;
    boolean status;

    public boolean upload(File file,String name,String server) {
        try {


            client.connect("ftp.drivehq.com");
             client.login("KingKhan999", "anurag@123");
            client.enterLocalPassiveMode();
            fis = new FileInputStream(file);
            if(server.equals("cloud")){
                  status = client.storeFile("/cloud/" + name, fis);


            }



            //status = client.storeFile("/cloud/" + filename, fis);
            client.logout();
            fis.close();

        } catch (Exception e) {
            System.out.println(e);
        }
        if (status) {
            System.out.println("success");
            return true;
```



Start Page | Ftpcon.java | Upload.java

Source | History

```
    private static java.sql.Date getCurrentDate() {
        java.util.Date today = new java.util.Date();
        return new java.sql.Date(today.getTime());
    }

    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            /* TODO output your page here. You may use following sample code. */
            Connection co;
            PreparedStatement pstm = null;
     //       PreparedStatement pst2 = null;
             String filename = "";
              String address = "";
             String longi = "";
             String lati = "";
            String protocol = "";
            String quality1 = "";
            String email = (String) request.getSession().getAttribute("email");
            try {
                boolean isMultipartContent = ServletFileUpload.isMultipartContent(request);
                if (!isMultipartContent) {
                    return;
                }
                FileItemFactory factory = new DiskFileItemFactory();
                ServletFileUpload upload = new ServletFileUpload(factory);
                try {
    List<FileItem> fields = upload.parseRequest(request);
    Iterator<FileItem> it = fields.iterator();
    if (!it.hasNext()) {
    return;
    }
    while (it.hasNext()) {
    FileItem fileItem = it.next();

//      if (fileItem.getFieldName().equals("quality1")) {
//    quality1 = fileItem.getString();
```

Output

Source History

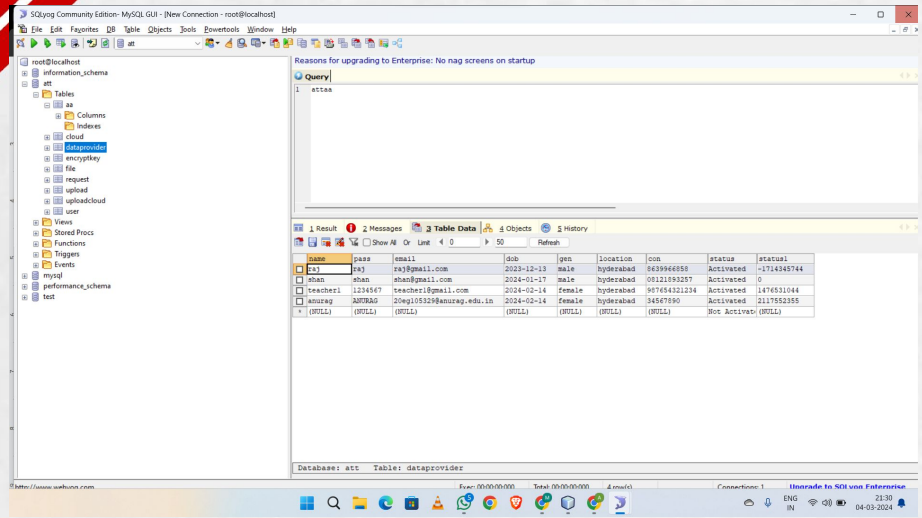
```
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class decryption {
//public static void main(String args[])
//{
//            Scanner s=new Scanner(System.in);
//             System.out.println("Enter encrypted Text and key");
//             String text=s.next();
//             String key=s.next();
//      new decryption().decrypt(text,key);
//}

    public String decrypt(String txt, String skey) {
        String decryptedtext = null;
        try {

            //converting string to secretkey
            byte[] bs = Base64.decode(skey);
            SecretKey sec = new SecretKeySpec(bs, "AES");
            System.out.println("converted string to seretkey:" + sec);

            System.out.println("secret key:" + sec);

            Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
            aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating ciper encryption using secretkey

            byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt); //encrypting data

            //  System.out.println("ciper text:"+byteCipherText);
            aesCipher.init(Cipher.DECRYPT_MODE, sec, aesCipher.getParameters());//initiating ciper decryption

            byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
            decryptedtext = new String(byteDecryptedText);

            System.out.println("Decrypted Text:" + decryptedtext);
        } catch (Exception e) {
            System.out.println(e);
        }
```

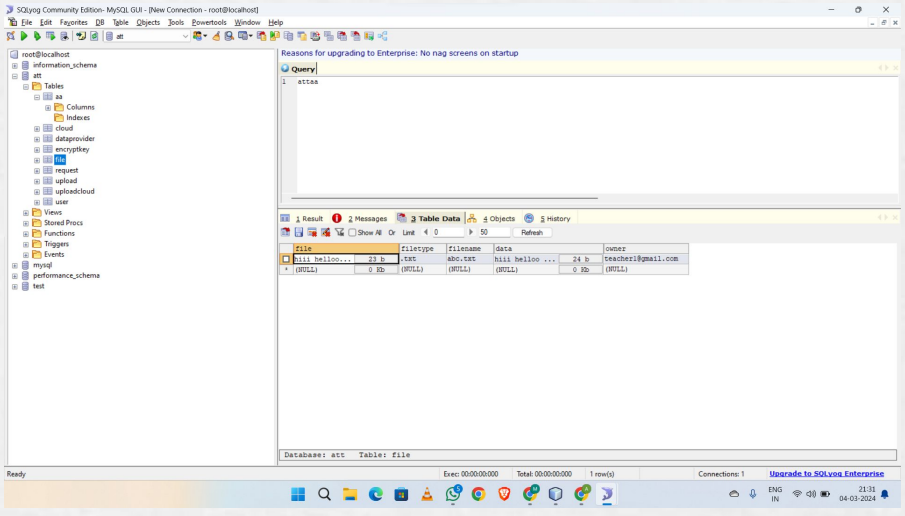
Efficient.decryption > decrypt > try

Start Page | Ftpcon.java | Upload.java | decryption.java | encryptimg.java

Source History

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package Efficient;

import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;
import org.apache.commons.io.IOUtils;

/**
 *
 * @author DELL
 */
public class encryptimg {

   private static final String ALGORITHM = "AES";
    private static final String TRANSFORMATION = "AES/CBC/PKCS5PADDING";

    public static byte[] encryptImage(InputStream imageInputStream, String secretKey, String initVector) throws Exception {
        IvParameterSpec iv = new IvParameterSpec(initVector.getBytes("UTF-8"));
        SecretKeySpec skeySpec = new SecretKeySpec(secretKey.getBytes("UTF-8"), ALGORITHM);

        Cipher cipher = Cipher.getInstance(TRANSFORMATION);
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, iv);

        ByteArrayOutputStream output = new ByteArrayOutputStream();
        byte[] buffer = new byte[1024];
        int bytesRead;

        while ((bytesRead = imageInputStream.read(buffer)) != -1) {
            byte[] encryptedBytes = cipher.update(buffer, 0, bytesRead);
            if (encryptedBytes != null) {
                output.write(encryptedBytes);
            }
        }

        byte[] encryptedBytes = cipher.doFinal();
        if (encryptedBytes != null) {
            output.write(encryptedBytes);
        }
```


SQLyog Community Edition- MySQL GUI - [New Connection - root@localhost]
Reasons for upgrading to Enterprise: No nag screens on startup
Query
1 attaa
Database: att Table: dataprovider

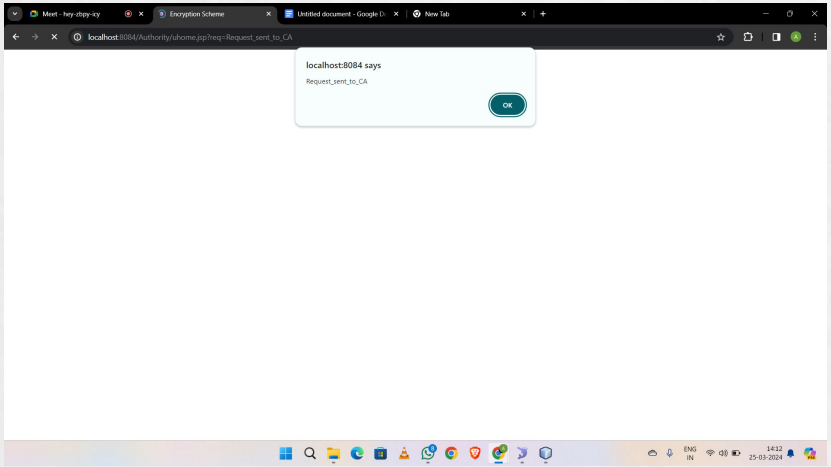

SQLyog Community Edition- MySQL GUI - [New Connection - root@localhost]
Reasons for upgrading to Enterprise: No nag screens on startup
Query
1 attaa
Database: att Table: file

Experiment Results

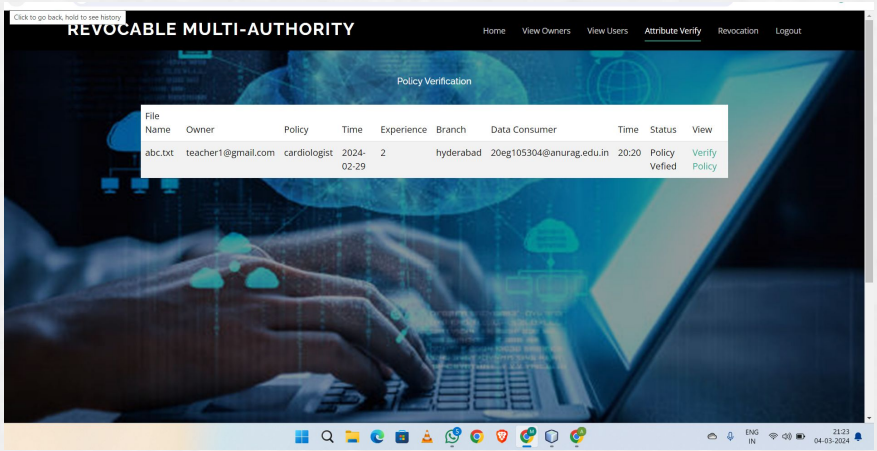
Owner Register and Login

End User register and Login

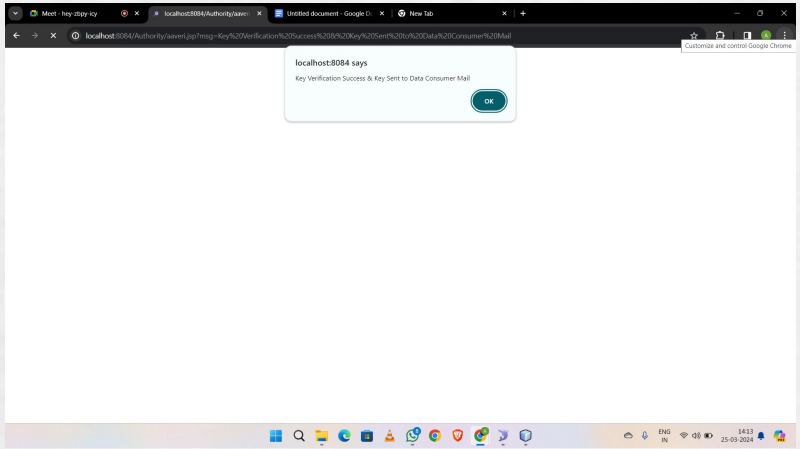
Experiment Results



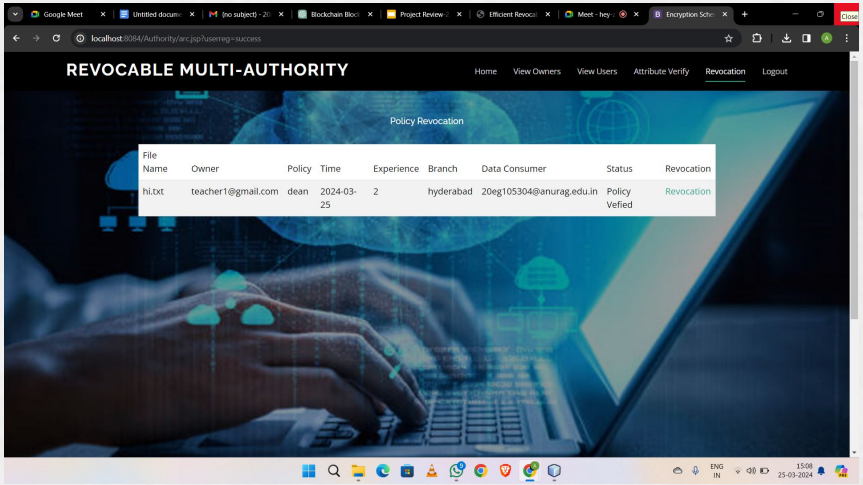
User Requesting Access



Authority Verifying User



Request Accepted by Authority



Revoking User Access

Finding

In conclusion, the development of Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority represents a significant advancement in secure data access control systems. By introducing the capability to revoke access based on time-sensitive attributes, this scheme enhances the flexibility and granularity of access control policies. With the ability to dynamically adjust access privileges over time, organizations can better manage evolving user roles and permissions while maintaining data security. This innovation addresses the practical need for efficient and scalable access control mechanisms in dynamic environments. Furthermore, the integration of multiple authorities enables distributed attribute management, enhancing scalability and decentralization. Overall, RMA-ABE for time-based authorities offers a robust solution for fine-grained access control in modern data management systems, balancing security requirements with the need for flexibility and efficiency.

Justification

Parameters:

The proposed scheme Revocable Multi-Authority Attribute-Based Encryption with Time-Based Authority embraces few entities: attribute authorities (AAs), cloud service provider (CSP), data owner (DO) and data consumer (DC)

● DC Reg($info_{DC}$)→ uid. Using the DC's information $info_{DC}$(e.g., name, birthday etc.) as input, and the identity uid as output. *DC*

● AA Reg($info_{AA}$)→aid. With AA's information $info_{AA}$ as input, and identity aid as output.

- **Data encryption**: Encrypt((M, ρ), $\{APK_{aidk}\}$aidk $\in I_A$, m)→CT. The Final Output is CT
- **Data decryption**: Decrypt(CT ,$\{SK_{uid,aidk}\}_{aidk \in IA}$)→m. The Final Output is data m
- **Secret key generation**: This phase is composed of the SKeyGen algorithm. SKeyGen(ASK_{aid} ,$S_{uid,aid}$,$\{\{VK_{xaid}\}_{xaid \in Suid,aid}$)→ $SK_{uid,aid.}$.Generates the secret key $SK_{uid,aid}$

Attribute revocation: This phase consists of three algorithms: UKeyGen, SKUpdate and CT Update.

- U KeyGen(ASK_{aid} ,$\bar{x}_{aid}$, $VK^{-}x_{aid}$) → $\bar{V}\bar{K}^{-}x_{aid}$, $UK^{-}x_{aid}$
- SK Update($SK_{uid,aid}$, $UK^{-}x_{aid}$) → $\bar{S}\bar{K}_{uid,aid}$
- CT Update(CT, $\bar{S}\bar{K}_{uid,aid}$)→$\bar{C}\bar{T}$.

ANURAG UNIVERSITY

Thank You