# 上海电力大学

# 实验报告

课程名称 信息安全概论 实验项目 上机实验2：HMAC算法和访问控制

姓名 洪勇 学号 20181389 班级 2018053

专业 计算机科学与技术 教师姓名 刘辉 实验日期 20200605

**一、实验目的和要求：**

理解HMAC算法与实施；

理解访问控制技术的基本原理；

掌握访问控制矩阵的实现原理与步骤

**二、实验内容：**

（1）访问控制矩阵的原理与实施过程；结合HMAC算法代码，结合实例，使用HMAC算法计算消息鉴别码

（2）访问控制矩阵的原理与实施过程；访问控制矩阵的实现

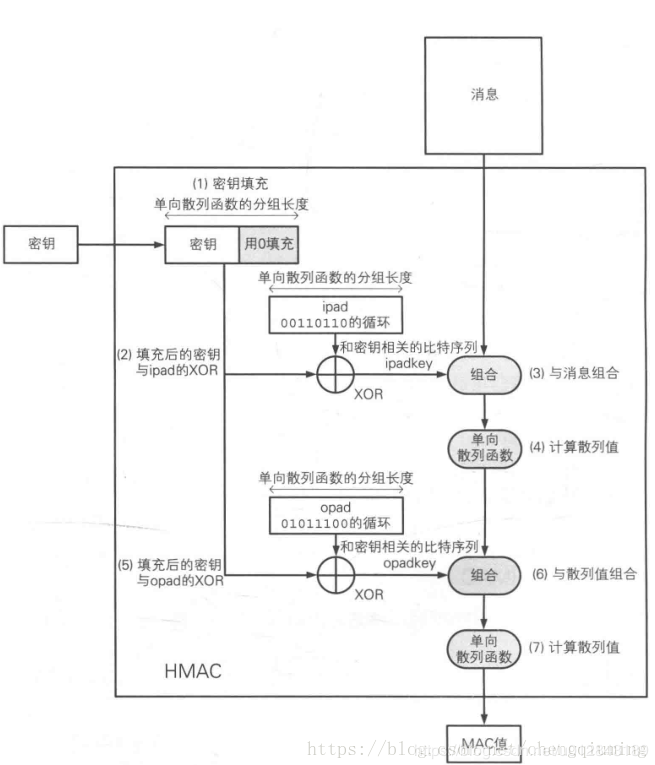
**三、实验步骤**

**1．对字符串“学号+姓名”，使用HMAC算法进行计算机消息鉴别码。**

**代码：**

**import** javax.crypto.KeyGenerator;  
**import** javax.crypto.Mac;  
**import** javax.crypto.SecretKey;  
**import** java.io.UnsupportedEncodingException;  
**import** java.security.InvalidKeyException;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.util.Base64;  
  
**public class** HMACSHA1{  
 **public static** SecretKey getSecretKey() **throws** NoSuchAlgorithmException {  
 KeyGenerator keyGenerator = KeyGenerator.*getInstance*(**"HmacMD5"**);  
 **return** keyGenerator.generateKey();  
 }  
 **public static void** main(String[] args) **throws** NoSuchAlgorithmException, InvalidKeyException, UnsupportedEncodingException {  
 SecretKey secretKey = *getSecretKey*();  
 Mac mac = Mac.*getInstance*(secretKey.getAlgorithm());  
 mac.init(secretKey);  
 String str = **"20181389洪勇"**;  
 **byte**[] utf8s = str.getBytes(**"UTF8"**);  
 **byte**[] doFinal = mac.doFinal(utf8s);  
 String encodeToString = Base64.*getEncoder*().encodeToString(doFinal);  
 System.***out***.println(encodeToString);  
 }  
}

**流程图：**



**结果：**



**2．模拟实现访问控制矩阵。**

**代码**

**FileOpera.class:**

**package** Matrix.DataFrame.element;  
  
**import** java.util.HashMap;  
**import** java.util.Map;  
**import** java.util.Set;  
  
**public class** FileOpera {  
 **private** String **fileName**;  
 **private** Map<String, Boolean> **right**;  
  
  
 **public** FileOpera(String fileName) {  
 **right** = **new** HashMap<String, Boolean>();  
 **right**.put(**"完全控制"**, **false**);  
 **right**.put(**"修改"**, **false**);  
 **right**.put(**"读取和执行"**, **false**);  
 **right**.put(**"读取"**, **false**);  
 **right**.put(**"写入"**, **false**);  
 **right**.put(**"特殊权限"**, **false**);  
 **this**.**fileName** = fileName;  
 }  
  
 **public** FileOpera() {  
 **this**(**""**);  
 }  
 @Override  
 **public** String toString() {  
 **return "FileOpera{"** +  
 **"fileName='"** + **fileName** + **'\''** +  
 **", right="** + **right** +  
 **'}'**;  
 }  
  
 **public** Map<String, Boolean> getRight() {  
 **return right**;  
 }  
  
 **public void** setRight(Map<String, Boolean> right) {  
 **this**.**right** = right;  
 }  
  
  
 **public** String getFileName() {  
 **return fileName**;  
 }  
  
 **public void** setFileName(String fileName) {  
 **this**.**fileName** = fileName;  
 }  
  
 **public boolean** changeRight(String rightName, **boolean** change){  
 Set<String> strings = **right**.keySet();  
 **if**(strings.contains(rightName)){  
 **right**.replace(rightName, change);  
 System.***out***.println(**"成功权限修改"**);  
 **return true**;  
 }  
 System.***out***.println(**"权限名不正确!"**);  
 **return false**;  
 }  
}

**DataRows.class:**

**package** Matrix.DataFrame;  
  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** DataRows{  
 **private** List<String> **rowElement**;  
  
 **public** DataRows() {  
 **rowElement** = **new** ArrayList<String>();  
 **rowElement**.add(**"/root/MYSQL"**);  
 **rowElement**.add(**"/root/T-SQL"**);  
 **rowElement**.add(**"/root/Tomcat"**);  
 **rowElement**.add(**"/root/C Plus Plus"**);  
 }  
  
 **public boolean** addFile(String fileName){  
 **if**(**rowElement** != **null**){  
 **rowElement**.add(fileName);  
 **return true**;  
 }**else return false**;  
 }  
  
 **public boolean** deleteFile(String fileName){  
 **for** (**int** i = 0; i < **rowElement**.size(); i++) {  
 **if**(**rowElement**.get(i).equals(fileName)){  
 **rowElement**.remove(i);  
 System.***out***.println(**"成功删除文件"**);  
 **return true**;  
 }  
 }  
 System.***err***.println(**"没有该文件"**);  
 **return false**;  
 }  
  
 **public** List<String> getRowElement() {  
 **return rowElement**;  
 }  
  
 **public void** setRowElement(List<String> rowElement) {  
 **this**.**rowElement** = rowElement;  
 }  
  
 **public void** display(){  
 **for** (**int** i = 0; i < **rowElement**.size(); i++) {  
 System.***out***.print(**rowElement**.get(i) + **"\t\t\t"**);  
 }  
 System.***out***.println();  
 }  
}

**User.class:**

**package** Matrix.DataFrame;  
  
**import** Matrix.DataFrame.element.FileOpera;  
  
**import** java.util.HashMap;  
**import** java.util.Map;  
**import** java.util.Set;  
  
**public class** User {  
 **private** String **username**;  
 **private** DataRows **head**;  
 **private** Map<String, FileOpera> **right**;  
  
 **public** Map<String, FileOpera> getRight() {  
 **return right**;  
 }  
  
 **public void** setRight(Map<String, FileOpera> right) {  
 **this**.**right** = right;  
 }  
  
 **public** User(String username, DataRows head) {  
 **this**.**username** = username;  
 **this**.**head** = head;  
 **right** = **new** HashMap<String, FileOpera>();  
 **for** (**int** i = 0; i < head.getRowElement().size(); i++) {  
 **right**.put(head.getRowElement().get(i), **new** FileOpera(head.getRowElement().get(i)));  
 }  
 }  
  
 **public** String getUsername() {  
 **return username**;  
 }  
  
 **public void** setUsername(String username) {  
 **this**.**username** = username;  
 }  
  
 **public** DataRows getHead() {  
 **return head**;  
 }  
  
 **public void** setHead(DataRows head) {  
 **this**.**head** = head;  
 updateRight(head);  
 System.***out***.println(**"更新完成"**);  
 }  
  
 **private void** updateRight(DataRows head){  
 HashMap<String, FileOpera> temp = **new** HashMap<String, FileOpera>();  
 **for** (**int** i = 0; i < head.getRowElement().size(); i++) {  
 **if**(**right**.containsKey(head.getRowElement().get(i))){  
 temp.put(head.getRowElement().get(i), **right**.get(head.getRowElement().get(i)));  
 }**else**{  
 temp.put(head.getRowElement().get(i), **new** FileOpera(head.getRowElement().get(i)));  
 }  
 }  
 **right** = temp;  
 }  
  
 **public void** display(){  
 Set<String> strings = **right**.keySet();  
 System.***out***.println(**username**);  
 **for** (String str :  
 strings) {  
 System.***out***.println(**right**.get(str).toString());  
 }  
 }  
  
 **public boolean** changeRight(String FileName, String rightName, **boolean** change){  
 Set<String> strings = **right**.keySet();  
 **if**(strings.contains(FileName)){  
 FileOpera fileOpera = **right**.get(FileName);  
 fileOpera.changeRight(rightName, change);  
 **right**.replace(FileName, fileOpera);  
 System.***out***.println(**"文件权限更新成功!"**);  
 **return true**;  
 }  
 System.***out***.println(**"文件权限更新失败"**);  
 **return false**;  
 }  
}

**DataColumns.class:**

**package** Matrix.DataFrame;  
  
**import** java.util.HashMap;  
**import** java.util.Map;  
**import** java.util.Set;  
  
**public class** DataColumns {  
 **private** Map<String, User> **columns**;  
  
 **public** DataColumns(DataRows head) {  
 **columns** = **new** HashMap<String, User>();  
 **columns**.put(**"root"**, **new** User(**"root"**, head));  
 **columns**.put(**"admin"**, **new** User(**"admin"**, head));  
 **columns**.put(**"guest"**, **new** User(**"guest"**, head));  
 }  
  
 **public** Map<String, User> getColumns() {  
 **return columns**;  
 }  
  
 **public void** setColumns(Map<String, User> columns) {  
 **this**.**columns** = columns;  
 }  
  
 **public void** dateUpdate(DataRows head){  
 Set<String> strings = **columns**.keySet();  
 **for** (String str :  
 strings) {  
 User user = **columns**.get(str);  
 user.setHead(head);  
 **columns**.replace(str, user);  
 }  
 }  
  
 **public void** display(){  
 Set<String> strings = **columns**.keySet();  
 **for** (String str :  
 strings) {  
 **columns**.get(str).display();  
 }  
 }  
  
 **public boolean** changeRight(String username, String Filename, String rightName, **boolean** change){  
 Set<String> strings = **columns**.keySet();  
 **if**(strings.contains(username)){  
 User user = **columns**.get(username);  
 user.changeRight(Filename, rightName, change);  
 **columns**.replace(username,user);  
 System.***out***.println(**"用户文件权限更改成功"**);  
 **return true**;  
 }  
 System.***out***.println(**"用户文件权限更改失败"**);  
 **return false**;  
 }  
  
 **public void** addUser(DataRows head, String username){  
 **if**(**columns**.containsKey(username)){  
 System.***out***.println(**"重名用户"**);  
 **return**;  
 }**else** {  
 **columns**.put(username, **new** User(username, head));  
 System.***out***.println(**"添加用户成功"**);  
 }  
 }  
}

**Matrix.class:**

**package** Matrix;  
  
**import** Matrix.DataFrame.DataColumns;  
**import** Matrix.DataFrame.DataRows;  
  
**public class** Matrix {  
 **private** DataRows **rows**;  
 **private** DataColumns **columns**;  
  
 **public** Matrix() {  
 **rows** = **new** DataRows();  
 **columns** = **new** DataColumns(**rows**);  
 }  
  
 **public** DataRows getRows() {  
 **return rows**;  
 }  
  
 **public void** setRows(DataRows rows) {  
 **this**.**rows** = rows;  
 }  
  
 **public** DataColumns getColumns() {  
 **return columns**;  
 }  
  
 **public void** setColumns(DataColumns columns) {  
 **this**.**columns** = columns;  
 }  
  
 **public void** display(){  
 **rows**.display();  
 **columns**.display();  
 }  
  
 **public void** addFile(String name){  
 **rows**.addFile(name);  
 **columns**.dateUpdate(**rows**);  
 }  
  
 **public void** deleteFile(String name){  
 **rows**.deleteFile(name);  
 **columns**.dateUpdate(**rows**);  
 }  
  
 **public void** changeRight(String username, String FileName, String rightName, **boolean** change){  
 **columns**.changeRight(username, FileName, rightName, change);  
 }  
  
 **public void** addUser(String username){  
 **columns**.addUser(**rows**, username);  
 }  
}

**demo.class:**

**package** Matrix;  
  
**import** org.junit.Test;  
  
**public class** demo {  
 **private static** Matrix *m*;  
  
 **public** Matrix getM() {  
 **return** *m*;  
 }  
  
 **public void** setM(Matrix m) {  
 **this**.*m* = m;  
 }  
  
 **public** demo() {  
 *m* = **new** Matrix();  
 }  
  
 **public static void** main(String[] args) {  
  
 }  
  
 @Test  
 **public void** testDisplay(){  
 *m*.display();  
 }  
  
 @Test  
 **public void** testAddFile(){  
 *m*.addFile(**"/root/idea"**);  
 *m*.display();  
 }  
  
 @Test  
 **public void** testDeletFile(){  
 *m*.addFile(**"/root/idea"**);  
 *m*.deleteFile(**"/root/idea"**);  
 *m*.display();  
 }  
  
 @Test  
 **public void** testDeleteFileNotIn(){  
 *m*.deleteFile(**"nmdwods"**);  
 *m*.display();  
 }  
  
 @Test  
 **public void** testChangedRight(){  
 *m*.changeRight(**"root"**, **"/root/MYSQL"**, **"完全控制"**, **true**);  
 *m*.display();  
 }  
  
 @Test  
 **public void** testAddUser(){  
 *m*.addUser(**"visitor"**);  
 *m*.display();  
 }  
}

**显示矩阵：**



**添加客体：**



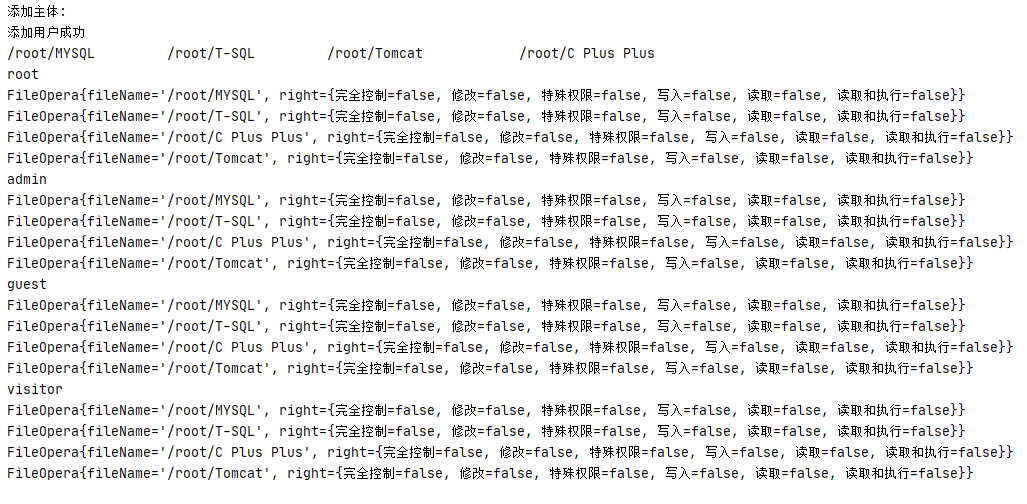
**删除客体：**



**更改文件的权限：**



**添加主体：**



**四、实验总结**

**由于第一个实现是找的代码，做起来也懵懵懂懂的。不过大体知道整个算法的流程。**

**控制矩阵是自己实现的，算是比较完整。实现了矩阵的客体添加，主体添加，能修改权限。实现了一个较为完整控制矩阵模型，能够动态的调整矩阵。**