Andrew De Stefano and Victoria Johnson

CMPT\_595\_01

Professor [Samanthula](https://montclair.instructure.com/courses/49825/users/59073)

3 December 2015

HMAC Project

Source Code

1. Signature Launcher

*/\*\*  
 \* Created by andrew on 11/5/15.  
 \*/***import** javax.xml.bind.DatatypeConverter;  
**import** java.io.FileReader;  
**import** java.io.BufferedReader;  
**import** java.security.MessageDigest;  
**import** java.io.FileNotFoundException;  
**import** java.io.IOException;  
**import** java.security.NoSuchAlgorithmException;  
**import** java.util.Scanner;  
  
**public class** SignatureLauncher  
{  
 */\*\*  
 \*  
 \** ***@param args*** *\*/* **public static void** main(String[] args)  
 {  
 *//Prompt user for password* System.out.println(**"Please enter your password to use the HMAC file verification program."**);  
 Scanner scanner = **new** Scanner(System.***in***);  
 String password = scanner.nextLine();  
 password = password.toLowerCase();  
 *//Get password as bytes* **byte**[] passwordBytes = password.getBytes();  
 *//Hash the password* **byte**[] hashedPasswordBytes = *getHashedPassword*(passwordBytes);  
 *//Compare user input against password file* **boolean** passwordMatch = *passwordCompare*(hashedPasswordBytes);  
 *//If password is good, prompt user for next step* **if**(passwordMatch == **true**){  
 System.out.println(**"Enter 'generate' to generate a hash value for a new file, or enter 'validate' to validate the existing files."**);  
 String input = scanner.nextLine();  
 input = input.toLowerCase();  
 *//Perform the signature step* **if**(input.equals(**"generate"**)){  
 Signature sig = **new** Signature(**"./directory1"**, **"./directory2"**, password);  
 sig.signatureAlgorithm();  
 *//Perform the validation step* } **else if**(input.equals(**"validate"**)){  
 Validator val = **new** Validator(**"./directory1"**, **"./directory2"**, password);  
 val.validationAlgorithm();  
 } **else**{  
 System.out.println(**"Invalid option, please try again."**);  
 }  
 } **else**{  
 System.out.println(**"Incorrect password, please try again."**);  
 }  
 }  
  
 */\*\*  
 \*  
 \** ***@param password*** *\** ***@return*** *\*/* **private static byte**[] getHashedPassword(**byte**[] password)  
 {  
 **byte**[] hashedPasswordBytes = **null**;  
 *//Storage for digest bytes* **byte**[] passwordFileBytes = **new byte**[password.**length**];  
 **try**{  
 *//Use SHA-512* MessageDigest hasher = MessageDigest.*getInstance*(**"SHA-512"**);  
 *//Hash the file bytes* hasher.update(passwordFileBytes);  
 *//Finish and cleanup* hashedPasswordBytes = hasher.digest();  
 hasher.reset();  
 } **catch**(NoSuchAlgorithmException exception){  
 System.out.println(exception.getMessage());  
 }  
 **return** hashedPasswordBytes;  
 }  
  
 */\*\*  
 \*  
 \** ***@param hashedPasswordBytes*** *\** ***@return*** *\*/* **private static boolean** passwordCompare(**byte**[] hashedPasswordBytes)  
 {  
 **boolean** match = **false**;  
 *//Convert the hash bytes to hex String* String hexHash1 = DatatypeConverter.*printHexBinary*(hashedPasswordBytes);  
 **try**{  
 *//Read password hex String from file* FileReader fileReader = **new** FileReader(**"./password.txt"**);  
 BufferedReader bufferedReader = **new** BufferedReader(fileReader);  
 **try**{  
 *//The hex String is all one line* String passwordFileContents = bufferedReader.readLine();  
 *//Check if the hex Strings of the two files are equal* **if**(hexHash1.equals(passwordFileContents)){  
 match = **true**;  
 }  
 } **catch**(IOException exception3){  
 System.out.println(exception3.toString());  
 }  
 } **catch**(FileNotFoundException exception2){  
 System.out.println(exception2.toString());  
 }  
 **return** match;  
 }  
}

2. Signature

*/\*\*  
 \* Created by andrew on 11/4/15.  
 \*/***import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
**import** java.io.FileNotFoundException;  
**import** java.io.PrintWriter;  
**import** java.security.InvalidKeyException;  
**import** java.security.NoSuchAlgorithmException;  
**import** javax.crypto.Mac;  
**import** javax.crypto.spec.SecretKeySpec;  
**import** javax.xml.bind.DatatypeConverter;  
  
**public class** Signature  
{  
 **private** String **path1**;  
 **private** String **path2**;  
 **private** String **password**;  
  
 */\*\*  
 \*  
 \** ***@param dir1Path*** *\** ***@param dir2Path*** *\** ***@param passwd*** *\*/* **public** Signature(String dir1Path, String dir2Path, String passwd)  
 {  
 **this**.**path1** = dir1Path;  
 **this**.**path2** = dir2Path;  
 **this**.**password** = passwd;  
 }  
  
 */\*\*  
 \*  
 \** ***@throws*** *NullPointerException  
 \*/* **public void** signatureAlgorithm() **throws** NullPointerException  
 {  
 *//Get directory1 as a File* File dir1 = **new** File(**this**.**path1**);  
 **try** {  
 *//Get all files from directory1* File[] dir1Files = dir1.listFiles();  
 *//Loop through all files* **for**(File file : dir1Files){  
 **try**{  
 *//Compute the digest of the file* **byte**[] bytes = **this**.messageDigest(file);  
 *//Write the digest to directory2* **this**.writeToOutFile(file, bytes);  
 } **catch**(NoSuchAlgorithmException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 }  
 } **catch**(SecurityException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 }  
  
 */\*\*  
 \*  
 \** ***@param file*** *\** ***@return*** *\** ***@throws*** *NoSuchAlgorithmException  
 \*/* **private byte**[] messageDigest(File file) **throws** NoSuchAlgorithmException  
 {  
 Mac hasher = Mac.*getInstance*(**"HmacSHA512"**);  
 SecretKeySpec hmacKey = **new** SecretKeySpec(**this**.**password**.getBytes(), **"HmacSHA512"**);  
 **try**{  
 hasher.init(hmacKey);  
 } **catch**(InvalidKeyException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 **byte**[] hashedBytes = **null**;  
 *//Storage for digest bytes* **byte**[] fileBytes = **new byte**[(**int**) file.length()];  
 **try**{  
 FileInputStream inputStream = **new** FileInputStream(file);  
 **try**{  
 *//Read fileBytes.length bytes from file* **int** bytesRead = inputStream.read(fileBytes);  
 inputStream.close();  
 *//Hash the file bytes* hashedBytes = hasher.doFinal(fileBytes);  
 hasher.reset();  
 } **catch**(IOException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 } **catch**(FileNotFoundException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 **return** hashedBytes;  
 }  
  
 */\*\*  
 \*  
 \** ***@param file*** *\** ***@param byteArray*** *\*/* **private void** writeToOutFile(File file, **byte**[] byteArray)  
 {  
 *//Fix the file extension of all hash files to be .txt* String extension = file.getName().substring(file.getName().lastIndexOf(**"."**));  
 String properFileName = file.getName().replace(extension, **".txt"**);  
 *//Create output file as concatenation of directory and filename* File outFile = **new** File(**this**.**path2**+**"/"**+properFileName);  
 *//Convert byte array to hexadecimal String* String hexHash = DatatypeConverter.*printHexBinary*(byteArray);  
 **try**{  
 *//Use a PrintWriter to write a String to outFile* PrintWriter printer = **new** PrintWriter(outFile);  
 printer.write(hexHash);  
 printer.close();  
 } **catch**(FileNotFoundException exception){  
 System.***out***.println(exception.toString());  
 }  
 }  
}

3. Validator

*/\*\*  
 \* Created by andrew on 11/4/15.  
 \*/***import** java.io.File;  
**import** java.io.FileInputStream;  
**import** java.io.IOException;  
**import** java.io.FileNotFoundException;  
**import** java.io.FileReader;  
**import** java.io.BufferedReader;  
**import** javax.crypto.Mac;  
**import** javax.crypto.spec.SecretKeySpec;  
**import** java.security.InvalidKeyException;  
**import** java.security.NoSuchAlgorithmException;  
**import** javax.xml.bind.DatatypeConverter;  
  
**public class** Validator  
{  
 **private** String **path1**;  
 **private** String **path2**;  
 **private** String **password**;  
  
 */\*\*  
 \*  
 \** ***@param dir1Path*** *\** ***@param dir2Path*** *\** ***@param passwd*** *\*/* **public** Validator(String dir1Path, String dir2Path, String passwd)  
 {  
 **this**.**path1** = dir1Path;  
 **this**.**path2** = dir2Path;  
 **this**.**password** = passwd;  
 }  
  
 */\*\*  
 \*  
 \*/* **public void** validationAlgorithm()  
 {  
 *//Get directory1 as a File* File dir1 = **new** File(**this**.**path1**);  
 *//Get directory2 as a File* File dir2 = **new** File(**this**.**path2**);  
 **try** {  
 *//Get all files from directory1 and 2* File[] dir1Files = dir1.listFiles();  
 File[] dir2Files = dir2.listFiles();  
 *//Loop through all files in directory1* **for**(File file1 : dir1Files){  
 *//For each file, get the file name  
 //Remove the file extension* String file1Name = file1.getName();  
 **int** index1 = file1Name.indexOf(**"."**);  
 String fixedFile1Name = file1Name.substring(0, index1);  
 *//Loop through all files in directory2* **for**(File file2: dir2Files){  
 *//Remove the file extension* String file2Name = file2.getName();  
 **int** index2 = file2Name.indexOf(**"."**);  
 String fixedFile2Name = file2Name.substring(0, index2);  
 *//Check if directory2 file name matches directory1 file name* **if**(fixedFile2Name.equals(fixedFile1Name)){  
 **this**.validate(file1, file2);  
 }  
 }  
 }  
 } **catch**(SecurityException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 }  
  
 */\*\*  
 \*  
 \** ***@param file1*** *\** ***@param file2*** *\*/* **private void** validate(File file1, File file2)  
 {  
 **byte**[] hashFile1 = **null**;  
 **try**{  
 *//Has file from directory1 again* hashFile1 = **this**.messageDigest(file1);  
 *//Convert the hash bytes to hex String* String hexHash1 = DatatypeConverter.*printHexBinary*(hashFile1);  
 **try**{  
 *//Read hex String from file in directory2* FileReader fileReader = **new** FileReader(file2);  
 BufferedReader bufferedReader = **new** BufferedReader(fileReader);  
 **try**{  
 *//The hex String is all one line* String file2Contents = bufferedReader.readLine();  
 *//Check if the hex Strings of the two files are equal* **if**(hexHash1.equals(file2Contents)){  
 System.***out***.println(file1.getName() + **" is valid"**);  
 } **else**{  
 System.***out***.println(file1.getName() + **"is NOT valid"**);  
   
 }  
 } **catch**(IOException exception3){  
 System.***out***.println(exception3.toString());  
 }  
 } **catch**(FileNotFoundException exception2){  
 System.***out***.println(exception2.toString());  
 }  
 } **catch**(NoSuchAlgorithmException exception){  
 System.***out***.println(exception.toString());  
 }  
 }  
  
 */\*\*  
 \*  
 \** ***@param file*** *\** ***@return*** *\** ***@throws*** *NoSuchAlgorithmException  
 \*/* **private byte**[] messageDigest(File file) **throws** NoSuchAlgorithmException  
 {  
 *//Use SHA-512* Mac hasher = Mac.*getInstance*(**"HmacSHA512"**);  
 SecretKeySpec hmacKey = **new** SecretKeySpec(**this**.**password**.getBytes(), **"HmacSHA512"**);  
 **try**{  
 hasher.init(hmacKey);  
 } **catch**(InvalidKeyException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 **byte**[] hashedBytes = **null**;  
 *//Storage for digest bytes* **byte**[] fileBytes = **new byte**[(**int**) file.length()];  
 **try**{  
 FileInputStream inputStream = **new** FileInputStream(file);  
 **try**{  
 *//Read fileBytes.length bytes from file* **int** bytesRead = inputStream.read(fileBytes);  
 inputStream.close();  
 *//Hash the file bytes* hashedBytes = hasher.doFinal(fileBytes);  
 hasher.reset();  
 } **catch**(IOException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 } **catch**(FileNotFoundException exception){  
 System.***out***.println(exception.getMessage());  
 }  
 **return** hashedBytes;  
 }  
}