Lab – Hmac

Q1.

Write a Java program (notes) to

- Get an instance of HmacSHA256
- Use it, along with a secret key, to create a HMAC signature for a message.
- Hash it again and check the equality of the two signatures.

Q2 & Q3.

We are going to simulate a sender (Q2) and receiver (Q3). Instead of sending directly, Q2 will put stuff into files and Q3 will read stuff from files.

Q2.

Write a Java program to

- Generate a HmacSHA256 secret key.
- Store it to file called “data/secretKey”
- Write a String message to “data/message.txt”
- Calculate the HMAC for the message and write it to the file “data/hmac”

Use the method

```
    static void writeToFile(String filename, Object object)
                                                    throws Exception {
    FileOutputStream fout = new FileOutputStream(filename);
    ObjectOutputStream oout = new ObjectOutputStream(fout);
    oout.writeObject(object);
    oout.close();
}
```

Q3.

Write a second Java program to

- Read the secret key from the file “data/secretKey”
- Read the text from the file “data/message.txt”
- Read the HMAC from the file “data/hmac”
- Calculate the HMAC for the message.
- Compare it with the value received.

Use the method

```
static Object readFromFile(String filename) throws Exception {
        FileInputStream fin = new FileInputStream(filename);
        ObjectInputStream oin = new ObjectInputStream(fin);
        Object object = oin.readObject();
        oin.close();
        return object;
}
```

Q4 & Q5.

Q4 is a sockets server that will send a challenge to the client. Q5 is a sockets client which reads the challenge, calculates the HMAC and sends it back to the server. The server then checks to see if the HMAC sent is correct. Complete the following two skeleton classes.

Q4.

```
public class H4ChallengeServer {

   static void writeToFile(String filename, Object object) throws Exception {
      FileOutputStream fout = new FileOutputStream(filename);
      ObjectOutputStream oout = new ObjectOutputStream(fout);
      oout.writeObject(object);
      oout.close();
   }

   public static void main(String[] args) {
      try {
         String challenge = "We give up, come and rule us";

         // generate secret and write to file

         // calculate the correct response to challenge

         // Accept connection from client
         Socket s;
         ServerSocket ss = new ServerSocket(2000);

         System.out.println("Server: waiting for connection ..");
         s = ss.accept();

         InputStream in = s.getInputStream();
         Scanner r = new Scanner(in);
         OutputStream o = s.getOutputStream();
         PrintWriter p = new PrintWriter(o);

         // read name and print it out
         String inputLine;
         inputLine = r.nextLine();
         System.out.println("Hello " + inputLine);

         // send challenge

         // read response from client

         // Compare the hmac received from the client with the hmac
calculated
         // If they match, send a message "success" to the client
         // If they don't, send a message "fails" to the client
```

```
            // close the connection
            p.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Q5.

```
public class H5ChallengeClient {

    static Object readFromFile(String filename) throws Exception {
        FileInputStream fin = new FileInputStream(filename);
        ObjectInputStream oin = new ObjectInputStream(fin);
        Object object = oin.readObject();
        oin.close();
        return object;
    }

    public static void main(String[] args) {

        try {
            // read secret key from the file

            // Make a connection to the server
            InetAddress inet = InetAddress.getByName("localhost");
            Socket s = new Socket(inet, 2000);
            OutputStream o = s.getOutputStream();
            PrintWriter p = new PrintWriter(o);
            InputStream in = s.getInputStream();
            Scanner r = new Scanner(in);

            // Send a name
            p.println("paul");
            p.flush();

            // read challenge from server

            // HMAC it using the secret key.

            // Base64 encode the hmac and send to the server

            // read reply and print it out
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```