

## Experiment 10

Write a Program to implement Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.

--

Roll No.	19
Name	Manav Jawrani
Class	D15-A
Subject	Security Lab
LO Mapped	LO1: To apply the knowledge of symmetric cryptography to implement classical ciphers. LO2 : Demonstrate Key management, distribution and user authentication.

**Aim:** Write a Program to implement Cryptographic Hash Functions and Applications (HMAC): to understand the need, design and applications of collision resistant hash functions.

### **Introduction:**

#### **What is HMAC?**

HMAC (Hash-based Message Authentication Code) is a type of a message authentication code (MAC) that is acquired by executing a cryptographic hash function on the data (that is) to be authenticated and a secret shared key. Like any of the MAC, it is used for both data integrity and authentication. Checking data integrity is necessary for the parties involved in communication. HTTPS, SFTP, FTPS, and other transfer protocols use HMAC. The cryptographic hash function may be MD-5, SHA-1, or SHA-256. Digital signatures are nearly similar to HMACs i.e they both employ a hash function and a shared key. The difference lies in the keys i.e HMACs use symmetric key(same copy) while Signatures use asymmetric (two different keys).

#### **Objectives:**

- As the Hash Function, HMAC is also aimed to be one way, i.e, easy to generate output from input but complex the other way round.
- It aims at being less affected by collisions than the hash functions.
- HMAC reuses the algorithms like MD5 and SHA-1 and checks to replace the embedded hash functions with more secure hash functions, in case found.
- HMAC tries to handle the Keys in a more simple manner.

#### **Applications:**

- Verification of e-mail address during activation or creation of an account.
- Authentication of form data that is sent to the client browser and then submitted back.
- HMACs can be used for Internet of things (IoT) due to less cost.
- Whenever there is a need to reset the password, a link that can be used once is sent without adding a server state.
- It can take a message of any length and convert it into a fixed-length message digest. That is even if you got a long message, the message digest will be small and thus permits maximizing bandwidth.

## Working of HMAC:

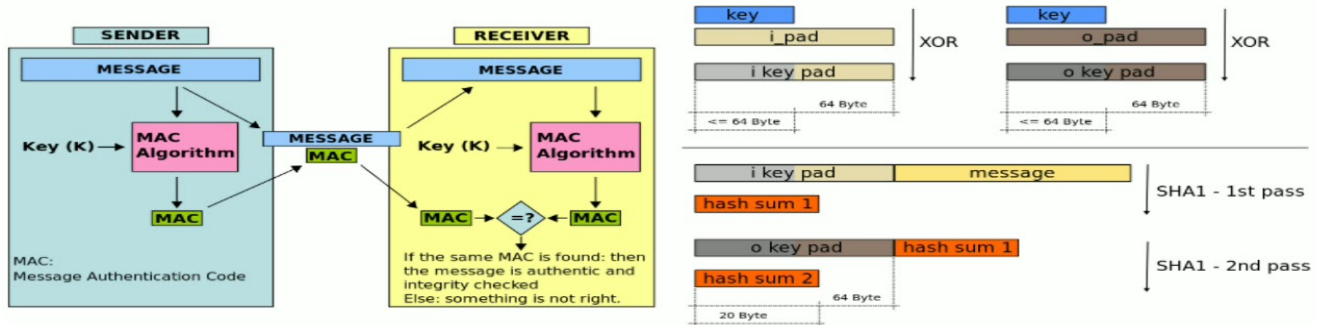
So basically, HMAC works this way. There is a Cryptographic Hash Function and Cryptographic

Hash Key. The HMAC is associated with a Hash function like SHA-256 or SHA-3 so the HMAC carries the name of the function used like HMAC-SHA-256 or HMAC-SHA-3. HMAC is a 2-phase computation. First, the secret key is used to derive 2 secret keys (inner and outer keys).

In the first phase, the algorithms produce an internal hash derived from the Message and inner key.

The second phases produce the final HMAC code derived from the inner hash and the outer key. This algorithm provides immunity against Length Extension Attacks. So, talking is cheap. Let's see how we do that in Java.

## MAC & HMAC (Wikipedia)



## Analysis of HMAC:

### Advantages:

- HMACs are ideal for high-performance systems like routers due to the use of hash functions which are calculated and verified quickly unlike the public key systems.
- Digital signatures are larger than HMACs, yet the HMACs provide comparably higher security.
- HMACs are used in administrations where public key systems are prohibited.

**Disadvantages:**

HMACs use shared keys which may lead to non-repudiation. If either sender or receiver's key is compromised then it will be easy for attackers to create unauthorized messages.

**Algorithm:**

1. Select K.
2. If  $K < b$ , pad 0's on left until  $k=b$ . K is between 0 and b ( $0 < K < b$ )
3. EXOR K+ with an ipad equivalent to b bits producing S1 bits.
4. Append S1 with plain text M
5. Apply SHA-512 on ( S1 || M )
6. Pad n-bits until length is equal to b-bits
7. EXOR K+ with opad equivalent to b bits producing S2 bits.
8. Append S2 with output of step 5.
9. Apply SHA-512 on step 7 to output n-bit hashcode.

**Implementation:****Code:**

```
import java.math.BigInteger;
import java.util.Base64;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

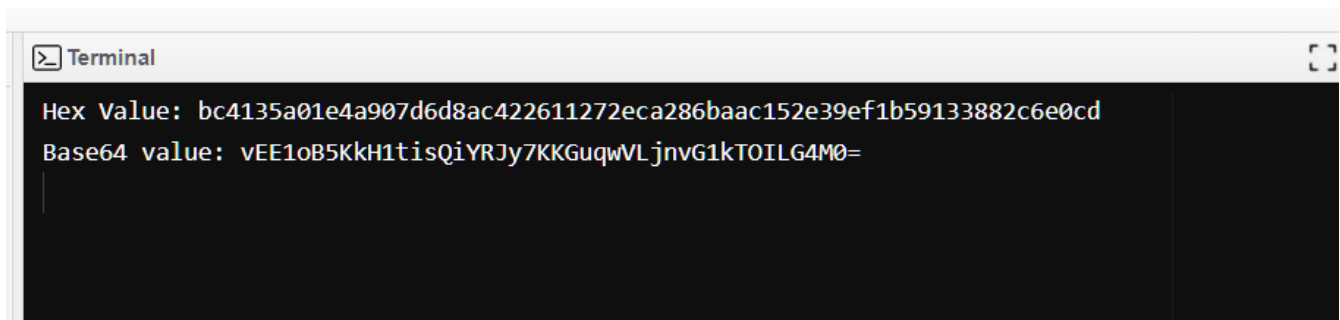
public class Main {
    private static class HMAC {
        public static byte[] hmac256(String secretKey,String message) {
            try {
                return hmac256(secretKey.getBytes("UTF-8"), message.getBytes("UTF-8"));
            } catch(Exception e) {
                throw new RuntimeException("Failed to generate HMACSHA256 encrypt",e);
            }
        }
        public static byte[] hmac256(byte[] secretKey,byte[] message) {
            byte[] hmac256 = null;
```

```
    try {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec sks = new SecretKeySpec(secretKey, "HmacSHA256");
        mac.init(sks);
        hmac256 = mac.doFinal(message);
        return hmac256;
    } catch (Exception e) {
        throw new RuntimeException("Failed to generate HMACSHA256 encrypt ");
    }
}

public static void main(String args[]) {
// passing plaintext in 2nd argument of hmac256
    byte[] hmacSha256 = HMAC.hmac256("secreT1_", "Hi Manav,How are you?");
    System.out.println(String.format("\nHex Value: %032x", new BigInteger(1,
        hmacSha256)));
    String base64HmacSha256 = Base64.getEncoder().encodeToString(hmacSha256);
    System.out.println("\nBase64 value: " + base64HmacSha256);
}
}
```

**Output:**

Message: Hi Manav,How are you?



```
Terminal
Hex Value: bc4135a01e4a907d6d8ac422611272eca286baac152e39ef1b59133882c6e0cd
Base64 value: vEE1oB5KkH1tisQiYRjy7KKGuqwVLjnvG1kTOILG4M0=
```

**Conclusion:**

Thus, we have successfully implemented Cryptographic Hash Functions and Applications (HMAC) and understood the need, design and applications of collision resistant hash functions.