**CNS Exp 7**

**Shashwat Tripathi  
D15A 64**

**Batch C**

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

**Theory:**

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).

Applications of HMAC:-

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.

HMACs provide clients and servers with a shared private key that is known only to them. The client makes a unique hash (HMAC) for every request. When the client requests the server, it hashes the requested data with a private key and sends it as a part of the request. Both the message and key are hashed in separate steps making it secure. When the server receives the request, it makes its own HMAC. Both the HMACS are compared and if both are equal, the client is considered legitimate.

**Code:**

Python

import hashlib

import hmac

import os

def generate\_salt(length=16):

# Generate a random salt for added security

return os.urandom(length)

def hash\_password(password, salt):

# Hash the password using SHA-256 and the provided salt

password\_hash = hashlib.pbkdf2\_hmac('sha256', password.encode('utf-8'), salt, 100000)

return password\_hash

def create\_hmac(key, message):

# Create an HMAC using SHA-256

h = hmac.new(key, message.encode('utf-8'), hashlib.sha256)

return h.digest()

def main():

# Example usage of cryptographic hash function

password = "secure\_password"

salt = generate\_salt()

hashed\_password = hash\_password(password, salt)

print("Password:", password)

print("Salt:", salt)

print("Hashed Password:", hashed\_password.hex())

# Example usage of HMAC

secret\_key = b'secret\_key'

message = "Hello, HMAC!"

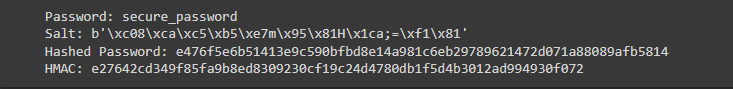
hmac\_result = create\_hmac(secret\_key, message)

print("HMAC:", hmac\_result.hex())

if \_\_name\_\_ == "\_\_main\_\_":

main()

**Output:**



**Conclusion:**

Hence, implemented Cryptographic Hash Functions and Applications (HMAC) and understood the design and applications of collision resistant hash functions.