# EX-NO-13-MESSAGE-AUTHENTICATION-CODE-MAC

#### AIM:

To implement MESSAGE AUTHENTICATION CODE(MAC)

#### **ALGORITHM:**

- 1. Message Authentication Code (MAC) is a cryptographic technique used to verify the integrity and authenticity of a message by using a secret key.
- 2. Initialization:
  - o Choose a cryptographic hash function (H) (e.g., SHA-256) and a secret key (K).
  - The message (M) to be authenticated is input along with the secret key (K).
- 3. MAC Generation:
  - Compute the MAC by applying the hash function to the combination of the message ( M ) and the secret key ( K ): [\text{MAC}(M, K) = H(K || M)] where ( || ) denotes concatenation of ( K ) and ( M ).
- 4. Verification:
  - The recipient, who knows the secret key ( K ), computes the MAC using the received message ( M ) and the same hash function.
  - The recipient compares the computed MAC with the received MAC. If they match, the message is authentic and unchanged.
- 5. Security: The security of the MAC relies on the secret key ( K ) and the strength of the hash function ( H ), ensuring that an attacker cannot forge a valid MAC without knowledge of the key.

## **Program:**

NAME: MUHAMMAD AFSHAN A REG NO: 212223100035 Q

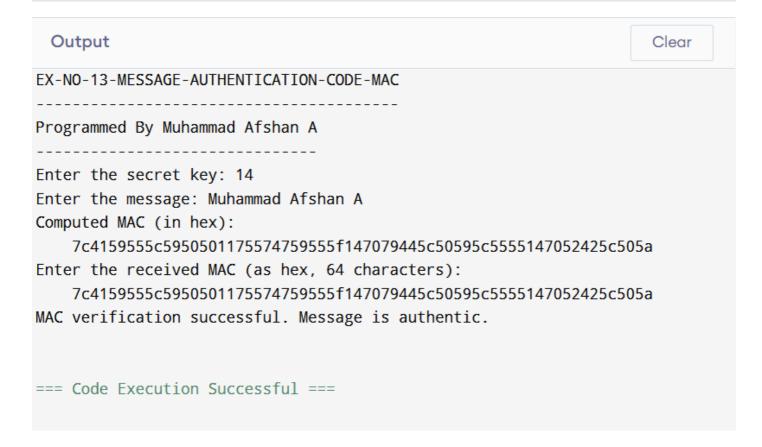
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

#define MAC\_SIZE 32 // Define MAC size in bytes

```
// Function to compute a simple MAC using XOR
void computeMAC(const char *key, const char *message, unsigned char *mac) {
    int key_len = strlen(key);
   int msg len = strlen(message);
   for (int i = 0; i < MAC_SIZE; i++) {
       mac[i] = key[i % key_len] ^ message[i % msg_len];
   }
}
int main() {
    printf("EX-NO-13-MESSAGE-AUTHENTICATION-CODE-MAC\n");
   printf("-----\n");
    printf("Programmed By Muhammad Afshan A\n");
   printf("-----\n");
   char key[100], message[100];
                                         // Buffer for computed MAC
   unsigned char mac[MAC_SIZE];
   unsigned char receivedMAC[MAC_SIZE]; // Buffer for received MAC
   char receivedHex[MAC_SIZE * 2 + 1];  // Buffer for received hex input (64 hex di
   // Step 1: Input secret key
   printf("Enter the secret key: ");
   scanf("%s", key);
   // Step 2: Input the message
   printf("Enter the message: ");
    scanf(" %[^\n]", message); // Reads message with spaces
   // Step 3: Compute the MAC
    computeMAC(key, message, mac);
   // Step 4: Display the computed MAC in hexadecimal
    printf("Computed MAC (in hex): ");
   for (int i = 0; i < MAC_SIZE; i++) {
       printf("%02x", mac[i]);
   printf("\n");
   // Step 5: Input the received MAC (as hex string)
   printf("Enter the received MAC (as hex, 64 characters): ");
   scanf("%64s", receivedHex);
   // Convert hex string to byte array
   for (int i = 0; i < MAC SIZE; i++) {
       if (sscanf(receivedHex + 2*i, "%2hhx", &receivedMAC[i]) != 1) {
           printf("Invalid hex input.\n");
           return 1;
   }
   // Step 6: Compare MACs
   if (memcmp(mac, receivedMAC, MAC SIZE) == 0) {
```

```
printf("MAC verification successful. Message is authentic.\n");
} else {
    printf("MAC verification failed. Message is not authentic.\n");
}
return 0;
}
```

## **Output:**



### Result:

The program is executed successfully.