21) Write a python program for CBC MAC of a oneblock message X, say T = MAC(K, X), the adversary immediately knows the CBC MAC for the two-block message X || (X ⊕ T) since this is once again.

PROGRAM:-

```
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad
import os
BLOCK_SIZE = 16  # AES block size in bytes
def xor_bytes(a, b):
    return bytes(x ^ y for x, y in zip(a, b))
def cbc_mac(key, message):
    cipher = AES.new(key, AES.MODE_ECB)
    prev = bytes([0]*BLOCK_SIZE)  # IV = 0
    for i in range(0, len(message), BLOCK_SIZE):
        block = message[i:i+BLOCK_SIZE]
        prev = cipher.encrypt(xor_bytes(block, prev))
    return prev
# Key generation
key = os.urandom(BLOCK_SIZE)
# One-block message
X = os.urandom(BLOCK_SIZE)
# CBC-MAC of X
T = cbc_mac(key, X)
print("Original message (X):", X.hex())
print("CBC-MAC T = MAC(K, X):", T.hex())
# Forged message: X || (X ⊕ T)
forged_block = xor_bytes(X, T)
forged_message = X + forged_block
# CBC-MAC of forged message
T_forged = cbc_mac(key, forged_message)
print("Forged message:       ", forged_message.hex())
print("CBC-MAC of forged message:", T_forged.hex())
```

```
# Check if the forged MAC is equal to T

print("Forgery successful?", T == T_forged)
```

OUTPUT:-

```
Original message (X): b329665092d6a0304c21de5924b0644c
CBC-MAC T = MAC(K, X): 8e6cb17c180bb8f503ba35f579931e8f
Forged message:         b329665092d6a0304c21de5924b0644c3d45d72c8add18c54f9bebac5d237ac3
CBC-MAC of forged message: 8e6cb17c180bb8f503ba35f579931e8f
Forgery successful? True
```
