Message Authentication Code (MAC)

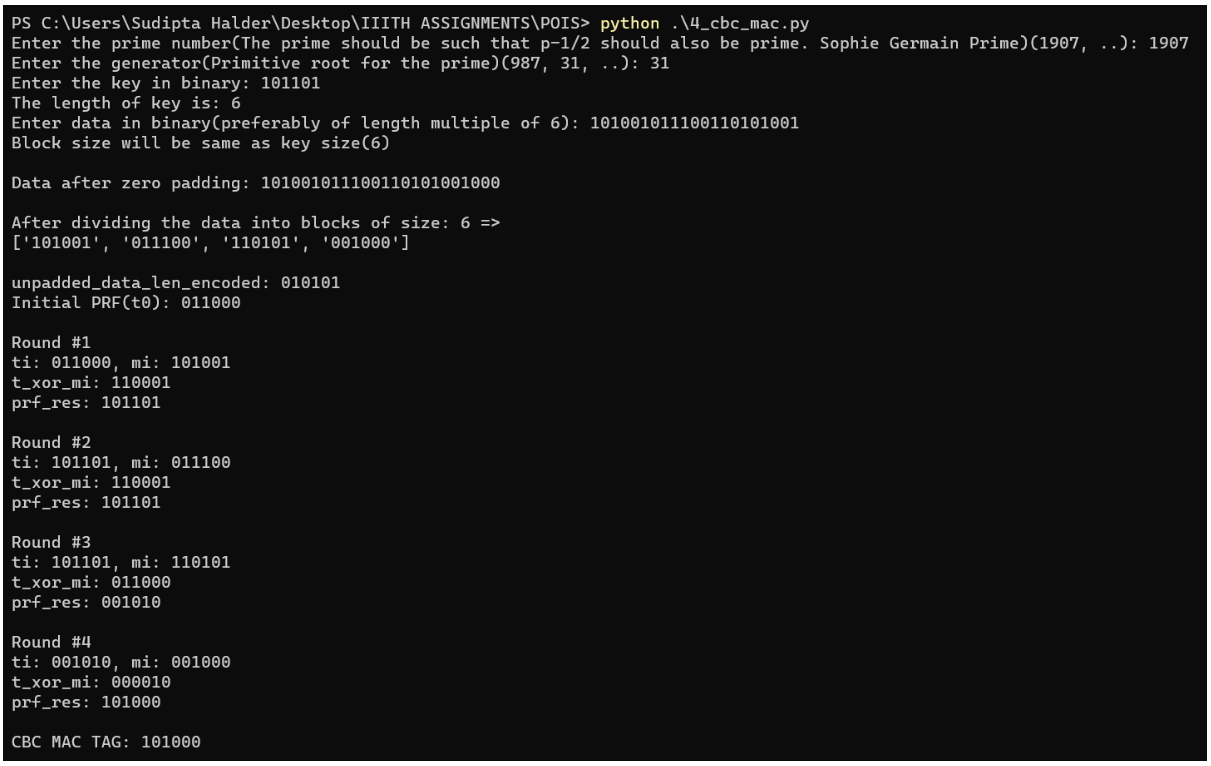
Assumptions: I've used **Cipher block chaining (CBC)** to design MAC.

Input format:

1. It will ask for a safe prime number p in integer format.
2. It will ask for a generator (primitive root) of that prime.
3. It will then ask for a key in binary form.
4. It will ask to enter the data in binary form.

Output Format:

1. It will output the CBC MAC TAG in binary format.

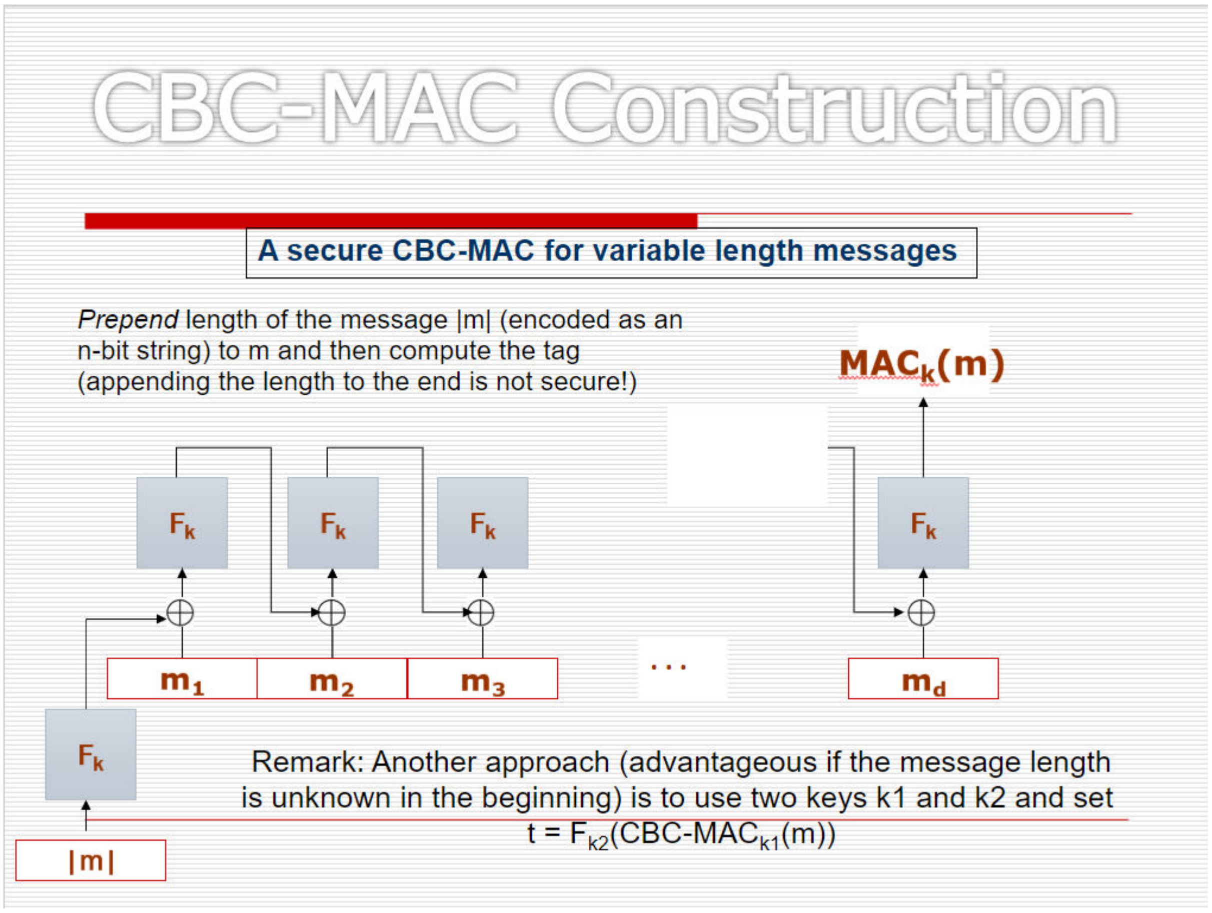


```
PS C:\Users\Sudipta Halder\Desktop\IIITH ASSIGNMENTS\POIS> python .\4_cbc_mac.py
Enter the prime number(The prime should be such that p-1/2 should also be prime. Sophie Germain Prime)(1907, ..): 1907
Enter the generator(Primitive root for the prime)(987, 31, ..): 31
Enter the key in binary: 101101
The length of key is: 6
Enter data in binary(preferably of length multiple of 6): 101001011100110101001
Block size will be same as key size(6)

Data after zero padding: 101001011100110101001000

After dividing the data into blocks of size: 6 =>
['101001', '011100', '110101', '001000']

unpadded_data_len_encoded: 010101
Initial PRF(t0): 011000

Round #1
ti: 011000, mi: 101001
t_xor_mi: 110001
prf_res: 101101

Round #2
ti: 101101, mi: 011100
t_xor_mi: 110001
prf_res: 101101

Round #3
ti: 101101, mi: 110101
t_xor_mi: 011000
prf_res: 001010

Round #4
ti: 001010, mi: 001000
t_xor_mi: 000010
prf_res: 101000

CBC MAC TAG: 101000
```

Working Flow:

1. Upon getting the input, the function `generate_cbc_mac(prime, generator, key, data, block_size)` will be called.
2. It will then check whether the data length is multiple of (block-size = key-size). If not, then it will pad zeros after the data to make the data length multiple of the key length.
3. Then, it will divide the data into blocks of block-size.
4. Then it will follow the CBC construction.
5. In first step, the data length will passed through the PRF. The obtained prf will be then xored with the first message block and passed through the PRF again.

6. For the next steps, the obtained prf from the previous iteration and the corresponsing msg block will be xored and then passed through PRF. Refer to the below diagram for better understanding.
7. In this way, the last output from the PRF will be used as MAC TAG.



CBC-MAC Construction

A secure CBC-MAC for variable length messages

Prepend length of the message |m| (encoded as an n-bit string) to m and then compute the tag (appending the length to the end is not secure!)

Remark: Another approach (advantageous if the message length is unknown in the beginning) is to use two keys k1 and k2 and set $t = F_{k2}(\text{CBC-MAC}_{k1}(m))$