# Chosen-ciphertext Attack (CCA)

**Assumptions:** I've used **output feedback mode (OFB)** to design CPA and **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 input the block length.
4. It will then ask for two keys k1 and k2 in binary form of block length.
5. It will ask to enter the data in binary form.
6. It will also ask for seed for generating initial vector via PRG.

**Output Format:**

1. It will first generate the ciphertext by OFB mode and CPA.
2. Then it will generate the MAC TAG by CBC mode on the ciphertext.
3. On the verification end, the MAC TAG will be verified first.
4. If the MAC matches, then the ciphertext will be decrypted to see if we get back the actual data. If yes, then it is successful.

```
PS C:\Users\Sudipta Halder\Desktop\IIITH ASSIGNMENTS\POIS> python .\5_cca_encryption.py
Enter the prime number(The prime should be such that p-1/2 should also be prime. Sophie Germain Prime)(1907): 31
Enter the generator(Primitive root for the prime)(987): 31
Enter (block_size = key_size) for the data: 4
Enter the key k1 in binary of length 4: 1001
Enter the key k2 in binary of length pref diff from k1 4: 1010
Enter the data to in binary(preferably of length multiple of 4): 110001100100101001011001

**********************Encryption Stage**********************
['1100', '0110', '0100', '1010', '0101', '1001']
Input a seed in binary for generating initial vector in PRG: 101101111
Initial vector is: 1111
Round #1
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1100
Round #2
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0110
Round #3
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0100
Round #4
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1010
Round #5
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0101
Round #6
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1001
The encrypted data is: 001110011011010110100110
```

```
***********************CBC MAC generation stage************************
['0011', '1001', '1011', '0101', '1010', '0110']
unpadded_data_len_encoded: 11000
Initial PRF(t0): 1111

Round #1
ti: 1111, mi: 0011
t_xor_mi: 1100
prf_res: 1111

Round #2
ti: 1111, mi: 1001
t_xor_mi: 0110
prf_res: 1111

Round #3
ti: 1111, mi: 1011
t_xor_mi: 0100
prf_res: 1111

Round #4
ti: 1111, mi: 0101
t_xor_mi: 1010
prf_res: 1111

Round #5
ti: 1111, mi: 1010
t_xor_mi: 0101
prf_res: 1111

Round #6
ti: 1111, mi: 0110
t_xor_mi: 1001
prf_res: 1111
CBC-MAC: 1111
```

```
***********************Verification Stage***********************
Received CBC MAC TAG: 1111
Received cipher text: 001110011011010110100110

***********************CBC MAC generation stage***********************
['0011', '1001', '1011', '0101', '1010', '0110']
unpadded_data_len_encoded: 11000
Initial PRF(t0): 1111

Round #1
ti: 1111, mi: 0011
t_xor_mi: 1100
prf_res: 1111

Round #2
ti: 1111, mi: 1001
t_xor_mi: 0110
prf_res: 1111

Round #3
ti: 1111, mi: 1011
t_xor_mi: 0100
prf_res: 1111

Round #4
ti: 1111, mi: 0101
t_xor_mi: 1010
prf_res: 1111

Round #5
ti: 1111, mi: 1010
t_xor_mi: 0101
prf_res: 1111

Round #6
ti: 1111, mi: 0110
t_xor_mi: 1001
prf_res: 1111
CBC-MAC: 1111
Mac Tag is verified. Sucess
```

```
**********************Decryption Stage***********************
['0011', '1001', '1011', '0101', '1010', '0110']
Round #1
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0011
Round #2
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1001
Round #3
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1011
Round #4
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0101
Round #5
The Initial Vector is: 1111
The PRF is : 1111
The data is : 1010
Round #6
The Initial Vector is: 1111
The PRF is : 1111
The data is : 0110

Decrypted data we got after verifying: 110001100100101001011001

MSG and MAC TAG both verified. Successful
```

**Working Flow:**

1. Upon getting the input, the function `generate_cpa_ofb(prime, generator, data, block_size, key)` will be called.
2. It will generate the cipher text c using key k1.
3. Then the ciphertext will be passed to ` generate_cbc_mac(prime, generator, key, data, block_size) ` and it will give the MAC TAG t using key k2.
4. After that, `verify_mac_tag_encrypted_data(prime, generator, k1, k2, block_size, encrypted_data, initial_vector, cbc_mac_tag) ` will be called to verify the correctness of the tag.
5. Upon successful verification of the MAC tag, ` cpa_ofb_decryption(encrypted_data, initial_vector, block_size, prime, generator, key)` will be called to verify the ciphertext. It will be decrypted and matched with the original data to see the correctness.
6. Upon successful verification, we would respond ` MSG and MAC TAG both verified. Successful`.

***CONSTRUCTION 4.17 CCA-secure encryption.***

Define a CCA-secure encryption scheme as follows:

- $\mathsf{Gen}'(1^n)$: upon input $1^n$, choose $k_1, k_2 \leftarrow \{0,1\}^n$
- $\mathsf{Enc}'_k(m)$: upon input key $(k_1, k_2)$ and plaintext message $m$, compute $c = \mathsf{Enc}_{k_1}(m)$ and $t = \mathsf{Mac}_{k_2}(c)$ and output the pair $(c, t)$
- $\mathsf{Dec}'_k(c, t)$: upon input key $(k_1, k_2)$ and ciphertext $(c, t)$, first verify that $\mathsf{Vrfy}_{k_2}(c, t) = 1$. If yes, then output $\mathsf{Dec}_{k_1}(c)$; if no, then output $\perp$.