

# CTF 문제에 출제된 암호 공격

IT정보공학과 신명수

- DH key exchange man in the middle attack
- CBC mode Bit-Flipping Attack
- Wiener attack
- Padding oracle attack
- Chosen ciphertext attack
- Etc...

# 목차

1. DH Man In the Middle Attack
2. CBC Bit Flipping Attack

# 1. DH Man In the Middle Attack

1.1 Diffie-Hellman key Exchange

1.2 Man In the Middle Attack

# 1.1 Diffie-Hellman key exchange

- 두 통신 당사자 사이에 공개되어도 상관없는 정보를 교환함으로써 비밀키를 공유할 수 있는 프로토콜.
- 이산 대수 문제(Discrete Logarithm Problem, DLP)을 기반으로 함.  
 $y = g^x \bmod p$  에서  $g, y, p$ 를 알아도  $x$ 를 알기 어렵다. ( $p$ 는 소수)  $p$

# 1.1 Diffie-Hellman key exchange

- Alice와 Bob의 통신



# 1.1 Diffie-Hellman key exchange

1. Generator  $g$ 와 소수  $p$ 를 결정한다.



$g, p$



$g, p$

# 1.1 Diffie-Hellman key exchange

2. Alice는 개인키  $a$ 를 이용하여  $A = g^a \text{ mod } p$  를 계산한다.



$$A = g^a \text{ mod } p$$



$$g, p$$



# 1.1 Diffie-Hellman key exchange

3. Bob은 개인키  $b$ 를 이용하여  $B = g^b \text{ mod } p$  를 계산한다.



$$A = g^a \text{ mod } p$$



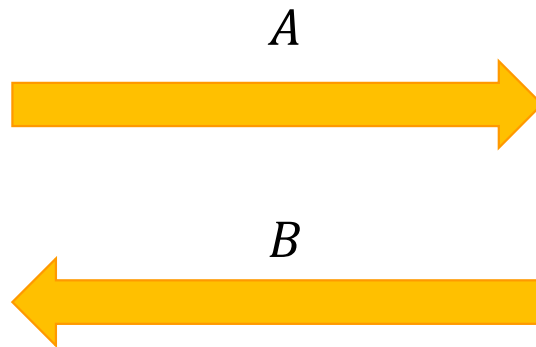
$$B = g^b \text{ mod } p$$

# 1.1 Diffie-Hellman key exchange

4. Alice는  $A$ 를 Bob에게 전송하고, Bob은  $B$ 를 Alice에게 전송한다.



$$A = g^a \bmod p$$



$$B = g^b \bmod p$$

# 1.1 Diffie-Hellman key exchange

5. 전송받은 값을 밑으로 하고 각자 개인키를 지수로 하는 거듭제곱연산한다.



$A$



$B$



$$A = g^a \bmod p$$

$$B^a \bmod p \equiv g^{ab} \bmod p$$

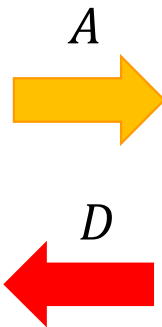
$$B = g^b \bmod p$$

$$A^b \bmod p \equiv g^{ab} \bmod p$$

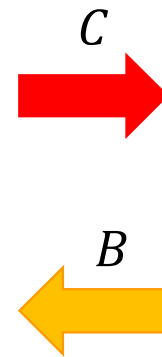
# 1.1 Diffie-Hellman key exchange

- Computational Diffie-Hellman Problem (CDH)  
비밀값  $a, b$  없이  $g^a, g^b$  만으로는  $g^{ab}$ 를 계산하는 문제.  
DLP를 풀 수 있다면 CDH 문제도 풀 수 있다.
- DLP는 적어도 CDH문제 만큼 난해하다.  
CDH문제가 적어도 DLP만큼 난해한지는 아직 증명되지 않음.

## 1.2 DH Man In the Middle Attack



$$A = g^a \text{ mod } p$$
$$D^a \text{ mod } p \equiv g^{ad} \text{ mod } p$$



$$C = g^c \text{ mod } p$$
$$D = g^d \text{ mod } p$$
$$A^d \text{ mod } p \equiv g^{ad} \text{ mod } p$$
$$B^c \text{ mod } p \equiv g^{bc} \text{ mod } p$$



$$B = g^b \text{ mod } p$$
$$C^b \text{ mod } p \equiv g^{bc} \text{ mod } p$$

## 1.2 DH Man In the Middle Attack

- Dreamhack:textbook-DH

Alice와 Bob의 통신을 도청중일 때 키 교환 과정을 공격해 flag를 획득해보자.

houma@

```
houma@ :~$ nc host3.dreamhack.games 16012
```

```
Prime: 0xd26f5cd1c2a59140bac2b4aed7ea8856b0f24b094a402d7b67efe65c2b470bf90e5e58e9bbe022f5f7567cfef89b03c8396b108da9f4f2e  
c39f8cbdef22d93f29da33c44ae262e9e58d1ed0d6385873c716e2ac34b326953741734e22a7979b98854d4a303544922a95beba33c754ffcce58cbe  
01031044b8eda5b16b291f469
```

```
Alice sends her key to Bob. Key: 0xc1cfd1189ebf17eb24b1b2538ae32dd9a0ff318207a216706b5a3ffdae0ae7427c972aaf9d8627379ab4a  
ef8a04bcae09d22548905c8a4952489ce11f333d350e609c4e474ba6527b74df89519207d549191b94e4766ffc48bbb8544b7424370ede6a77a0c420  
8ed7ab4459e41bd8ad940eca8dbe89055c96744e8061111cd96
```

```
Let's interrupt !
```

```
>>
```

## 1.2 DH Man In the Middle Attack

```
29 flag = open("flag", "r").read().encode()
30 prime = getPrime(1024)
31 print(f"Prime: {hex(prime)}")
32 alice = Person(prime)
33 bob = Person(prime)
34
35 alice_k = alice.calc_key()
36 print(f"Alice sends her key to Bob. Key: {hex(alice_k)}")
37 print("Let's interrupt !")
38 alice_k = int(input(">> "))
39 if alice_k == alice.g:
40     exit("Malicious key !!")
41 bob.set_shared_key(alice_k)
42
43 bob_k = bob.calc_key()
44 print(f"Bob sends his key to Alice. Key: {hex(bob_k)}")
45 print("Let's interrupt !")
46 bob_k = int(input(">> "))
47 if bob_k == bob.g:
48     exit("Malicious key !!")
49 alice.set_shared_key(bob_k)
50
51 print("They are sharing the part of flag")
52 print(f"Alice: {alice.encrypt(flag[:len(flag) // 2])}")
53 print(f"Bob: {bob.encrypt(flag[len(flag) // 2:])}")
```

```
8 class Person(object):
9     def __init__(self, p):
10         self.p = p
11         self.g = 2
12         self.x = random.randint(2, self.p - 1)
13
14     def calc_key(self):
15         self.k = pow(self.g, self.x, self.p)
16         return self.k
17
18     def set_shared_key(self, k):
19         self.sk = pow(k, self.x, self.p)
20         aes_key = hashlib.md5(str(self.sk).encode()).digest()
21         self.cipher = AES.new(aes_key, AES.MODE_ECB)
22
23     def encrypt(self, pt):
24         return self.cipher.encrypt(pad(pt, 16)).hex()
25
26     def decrypt(self, ct):
27         return unpad(self.cipher.decrypt(bytes.fromhex(ct)), 16)
```

## 1.2 DH Man In the Middle Attack

```
houma@ :~$ nc host3.dreamhack.games 16012
Prime: 0xd2b+5cd1c2a59140bac2b4aed7ea8856b0f24b094a402d7b67efe65c2b470bf90e5e58e9bbe022f5f7567cfe89b03c8396b108da9f4f2e
c39f8cbbedf22d93f29da33c44ae262e9e58d1ed0d6385873c716e2ac34b326953741734e22a7979b98854d4a303544922a95beba33c754ffcce58cbe
01031044b8eda5b16b291f469
Alice sends her key to Bob. Key: 0xc1cfd1189ebf17eb24b1b2538ae32dd9a0ff318207a216706b5a3ffdae0ae7427c972aaf9d8627379ab4a
ef8a04bcae09d22548905c8a4952489ce11f333d350e609c4e474ba6527b74df89519207d549191b94e4766ffc48bbb8544b7424370ede6a77a0c420
8ed7ab4459e41bd8ad940eca8dbe89055c96744e8061111cd96
Let's interrupt !
>> 4
Bob sends his key to Alice. Key: 0xcfc829b738c9cf9111f952cd8b23744ab7a9deb34daf81e5cc2e24df7ed2214801681e4215926f231280c7
c4ff0c08eeef21861f4e0555111561b4396eb1f9ef584acc447942496fe99065cd0a4253ea1decdb9db7c1e571232dd2c8af733606d4922c76ac63e8d
a98a36e23921989023c5688ee62d663d2cb29c331665c12dc05
Let's interrupt !
>> 8
They are sharing the part of flag
Alice: 29c3ee83956e131081cc2ad02a4b32c036b5e7c33e3f4866c7181ea85323d0cb3567fbbd7833d7305a9999a396e4d3b9
Bob: 840e4b9709df0b169f60ca552025751b55499769bd445989025724249c1aa6b0348a7702520210a2e5631ab118324716
```

$$g = 2$$

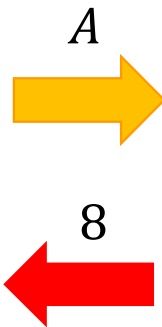
$$C = g^2 \text{ mod Prime} = 4$$

$$D = g^3 \text{ mod Prime} = 8$$

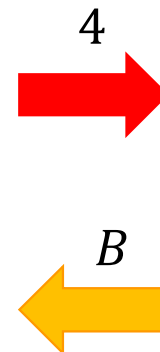


## 1.2 DH Man In the Middle Attack

$$c = 2, d = 3$$



$$A = 2^a \bmod p$$
$$8^a \bmod p \equiv 2^{3a} \bmod p$$



$$4 = 2^2 \bmod p$$
$$8 = 2^3 \bmod p$$
$$A^3 \bmod p \equiv 2^{3a} \bmod p$$
$$B^2 \bmod p \equiv 2^{2b} \bmod p$$



$$B = 2^b \bmod p$$
$$4^b \bmod p \equiv 2^{2b} \bmod p$$

## 1.2 DH Man In the Middle Attack

```
29 flag = open("flag", "r").read().encode()
30 prime = getPrime(1024)
31 print(f"Prime: {hex(prime)}")
32 alice = Person(prime)
33 bob = Person(prime)
34
35 alice_k = alice.calc_key()
36 print(f"Alice sends her key to Bob. Key: {hex(alice_k)}")
37 print("Let's interrupt !")
38 alice_k = int(input(">> "))
39 if alice_k == alice.g:
40     exit("Malicious key !!")
41 bob.set_shared_key(alice_k)
42
43 bob_k = bob.calc_key()
44 print(f"Bob sends his key to Alice. Key: {hex(bob_k)}")
45 print("Let's interrupt !")
46 bob_k = int(input(">> "))
47 if bob_k == bob.g:
48     exit("Malicious key !!")
49 alice.set_shared_key(bob_k)
50
51 print("They are sharing the part of flag")
52 print(f"Alice: {alice.encrypt(flag[:len(flag) // 2])}")
53 print(f"Bob: {bob.encrypt(flag[len(flag) // 2:])}")
```

```
8 class Person(object):
9     def __init__(self, p):
10         self.p = p
11         self.g = 2
12         self.x = random.randint(2, self.p - 1)
13
14     def calc_key(self):
15         self.k = pow(self.g, self.x, self.p)
16         return self.k
17
18     def set_shared_key(self, k):
19         self.sk = pow(k, self.x, self.p)
20         aes_key = hashlib.md5(str(self.sk).encode()).digest()
21         self.cipher = AES.new(aes_key, AES.MODE_ECB)
22
23     def encrypt(self, pt):
24         return self.cipher.encrypt(pad(pt, 16)).hex()
25
26     def decrypt(self, ct):
27         return unpad(self.cipher.decrypt(bytes.fromhex(ct)), 16)
```

## 1.2 DH Man In the Middle Attack

```

6 prime = 0xd26f5cd1c2a5914f0b6ac2b4aed7ea8856b0f24b54a402d7b67efe65c2b470bf90e5e58e9bbe022f5f7567cfeef89b03c8396b108d
7 A = 0xc1cfd1189ebf17eb24b1b2538ae32dd9a0ff318207a216706b5a3ffdae0ae7427c972aaf9d8627379ab4aef8a04bcae09d22548905c8
8 B = 0xcfc829b738c9cf9111f952cd8b23744ab7a9deb34daf81e5cc2e24df7ed2214801681e4215926f231280c7c4ff0c08eeef21861f4e05f
9
10
11 Alice_flag = '29c3ee83956e131081cc2ad02a4b32c036b5e7c33e3f4866c7181ea85323d0cb3567fbdd7833d7305a9999a396e4d3b9'
12 Bob_flag = '840e4b9709df0b169f60ca552025751b55499769bd445989025724249c1aa6b0348a7702520210a2e5631ab118324716'
13
14 c = 2; d = 3
15 Asharedkey = pow(A, d, prime)
16 Aaes_key = hashlib.md5(str(Asharedkey).encode()).digest()
17 Acipher = AES.new(Aaes_key, AES.MODE_ECB)
18
19 Bsharedkey = pow(B, c, prime)
20 Baes_key = hashlib.md5(str(Bsharedkey).encode()).digest()
21 Bcipher = AES.new(Baes_key, AES.MODE_ECB)
22
23 Aflag = Acipher.decrypt(bytes.fromhex(Alice_flag))
24 bflag = Bcipher.decrypt(bytes.fromhex(Bob_flag))
25
26 print(Aflag, '\n-----\n', 'MINGW64 /')
27 print(bflag, '$ python solution.py')

```

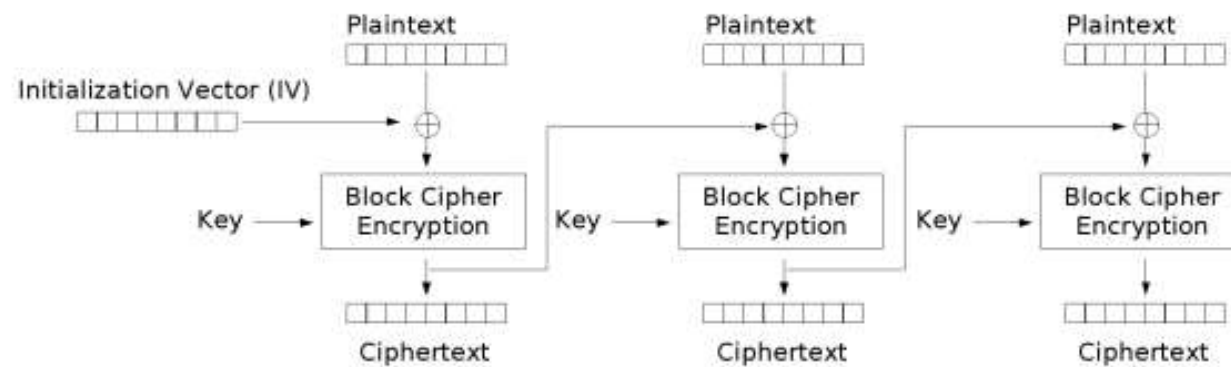
## 2. CBC Bit-Flipping Attack

2.1 CBC mode

2.2 CBC Bit-Flipping Attack

## 2.1 CBC mode

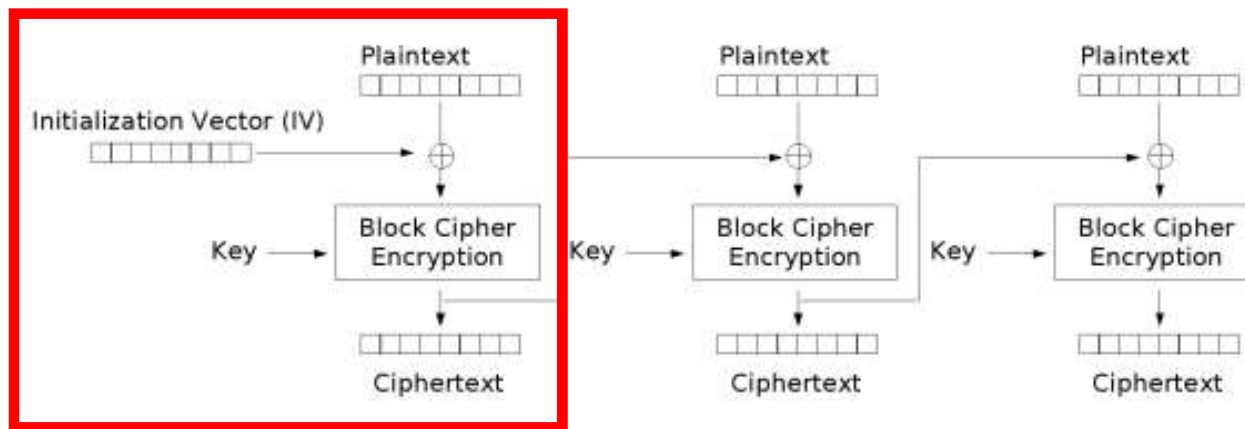
- Cipher Block chaining(CBC) mode



Cipher Block Chaining (CBC) mode encryption

## 2.1 CBC mode

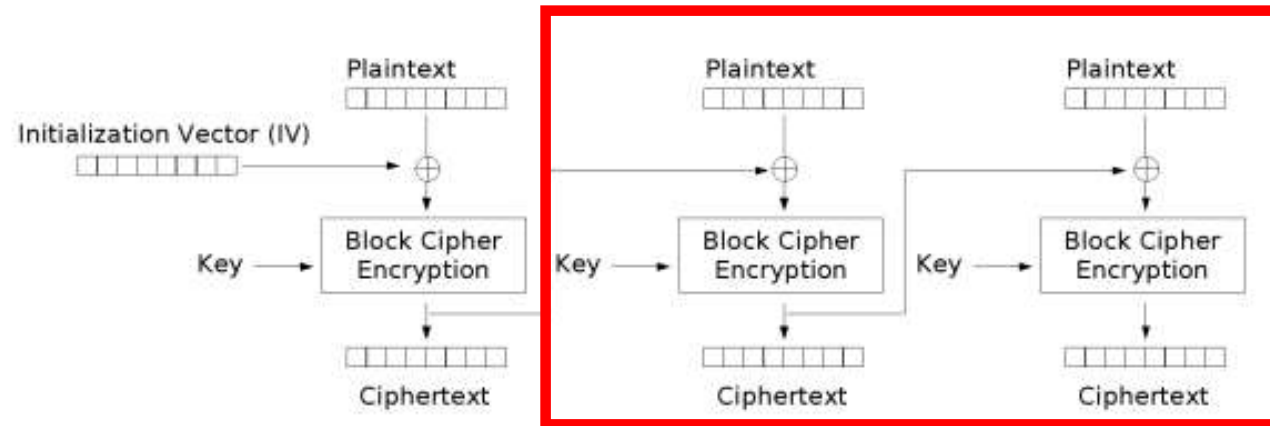
$$CT_0 = \text{Enc}(PT_0 \oplus IV, \text{key})$$



Cipher Block Chaining (CBC) mode encryption

## 2.1 CBC mode

$$CT_i = Enc(PT_i \oplus CT_{i-1}, key) \quad (1 \leq i)$$

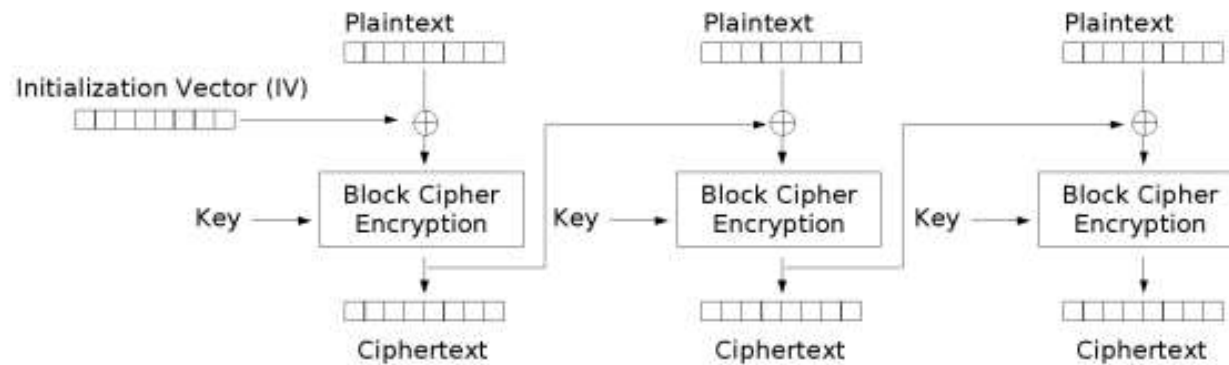


Cipher Block Chaining (CBC) mode encryption

## 2.1 CBC mode

$$CT_0 = \text{Enc}(PT_0 \oplus IV, \text{key})$$

$$CT_i = \text{Enc}(PT_i \oplus CT_{i-1}, \text{key}) \quad (1 \leq i)$$

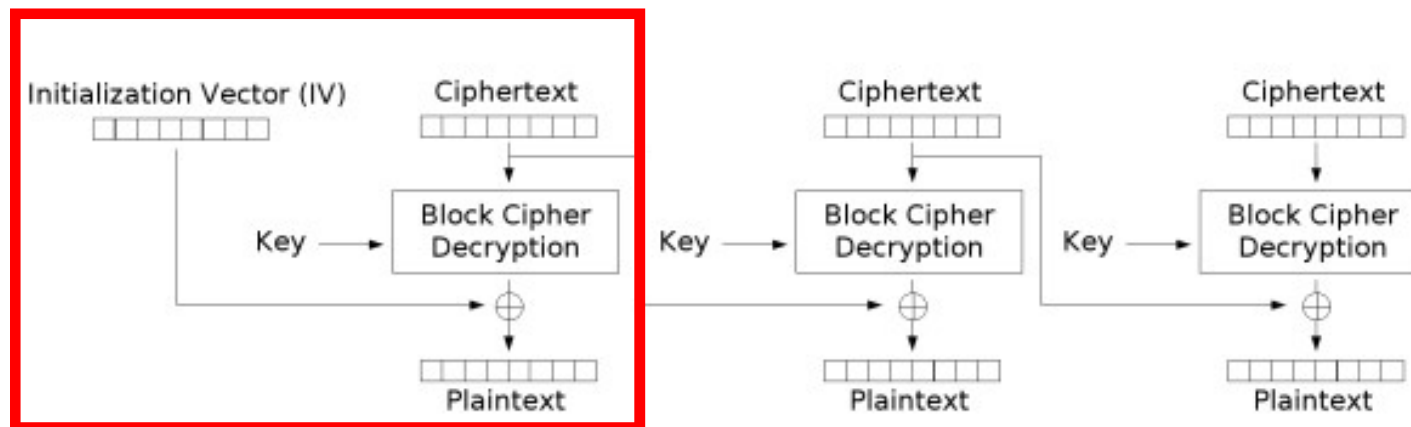


Cipher Block Chaining (CBC) mode encryption



## 2.1 CBC mode

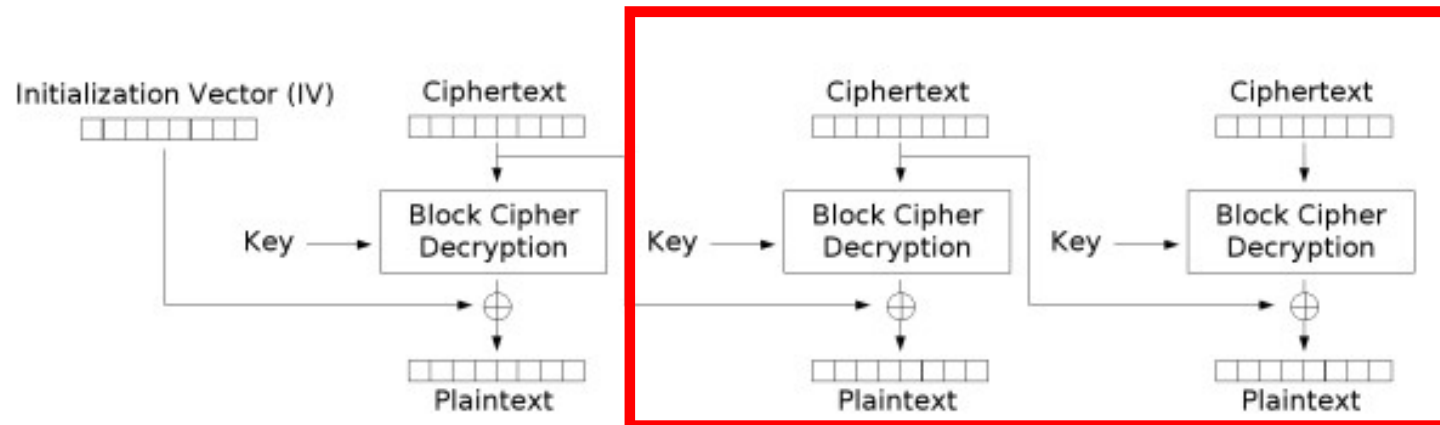
$$PT_0 = Dec(CT_0, key) \oplus IV$$



Cipher Block Chaining (CBC) mode decryption

## 2.1 CBC mode

$$PT_i = Dec(CT_i, key) \oplus CT_{i-1} \quad (1 \leq i)$$

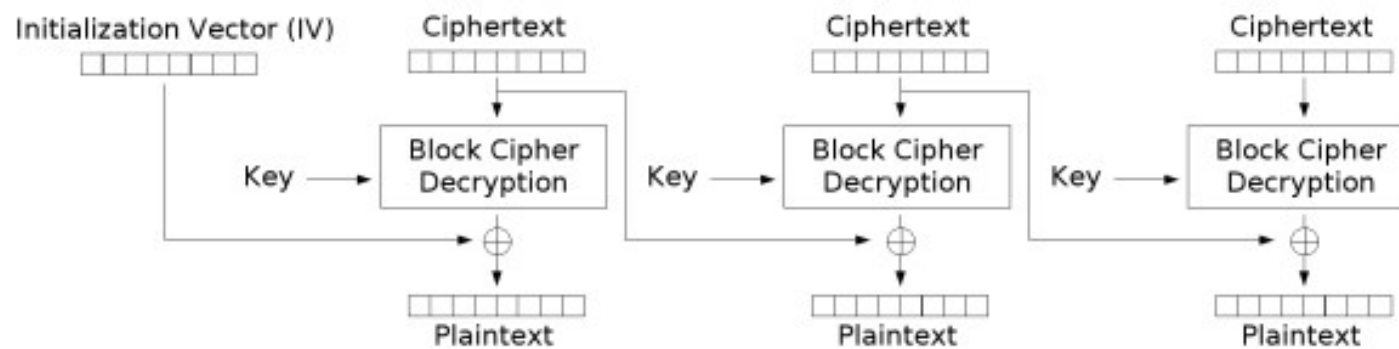


Cipher Block Chaining (CBC) mode decryption

## 2.1 CBC mode

$$PT_0 = Dec(CT_0, key) \oplus IV$$

$$PT_i = Dec(CT_i, key) \oplus CT_{i-1} \quad (1 \leq i)$$



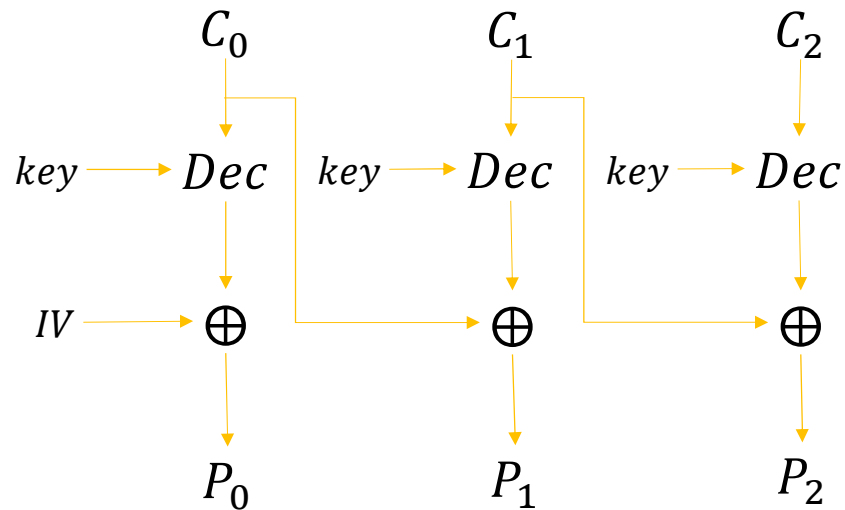
Cipher Block Chaining (CBC) mode decryption

## 2.2 CBC Bit-Flipping Attack

- 암호문을 수정하여 복호화된 평문에 예측가능한 영향을 끼치는 공격. 공격자는 메시지 복호화를 하지 않더라도 평문을 수정한다.
- 공격자가 평문의 형태를 알고 있다면, 원하는 부분을 수정할 수 있다.
- 암호는 인증이 아님을 입증하는 공격. 단순히 메시지를 암호화하는 것만으로는 충분하지 않다.

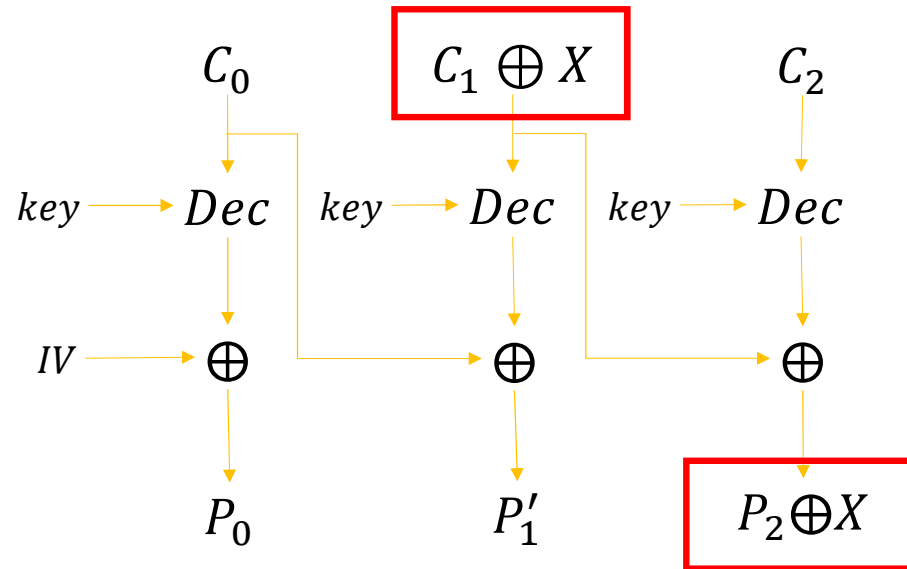
## 2.2 CBC Bit-Flipping Attack

- 복호화 과정



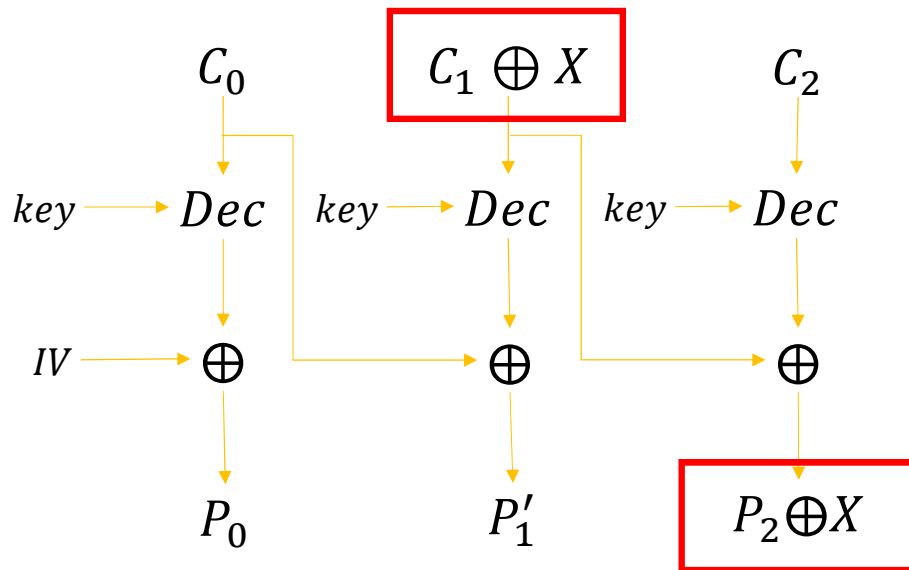
## 2.2 CBC Bit-Flipping Attack

- 복호화 과정



## 2.2 CBC Bit-Flipping Attack

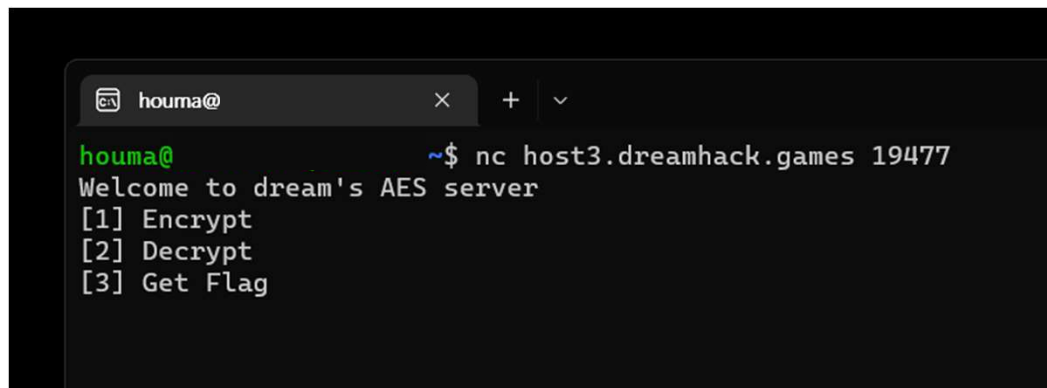
- 복호화 과정



$$P_2 = Dec(C_2, key) \oplus C_1$$
$$Dec(C_2, key) \oplus (C_1 \oplus X) = P_2 \oplus X$$

## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC

A terminal window with a dark background and light text. The window title bar shows 'houma@' and standard window controls. The terminal content shows a netcat listener on host3.dreamhack.games port 19477. It receives a connection from 'houma@' and displays a welcome message and a menu with three options: [1] Encrypt, [2] Decrypt, and [3] Get Flag.

```
houma@ ~$ nc host3.dreamhack.games 19477
Welcome to dream's AES server
[1] Encrypt
[2] Decrypt
[3] Get Flag
```



## 2.2 CBC Bit-Flipping Attack

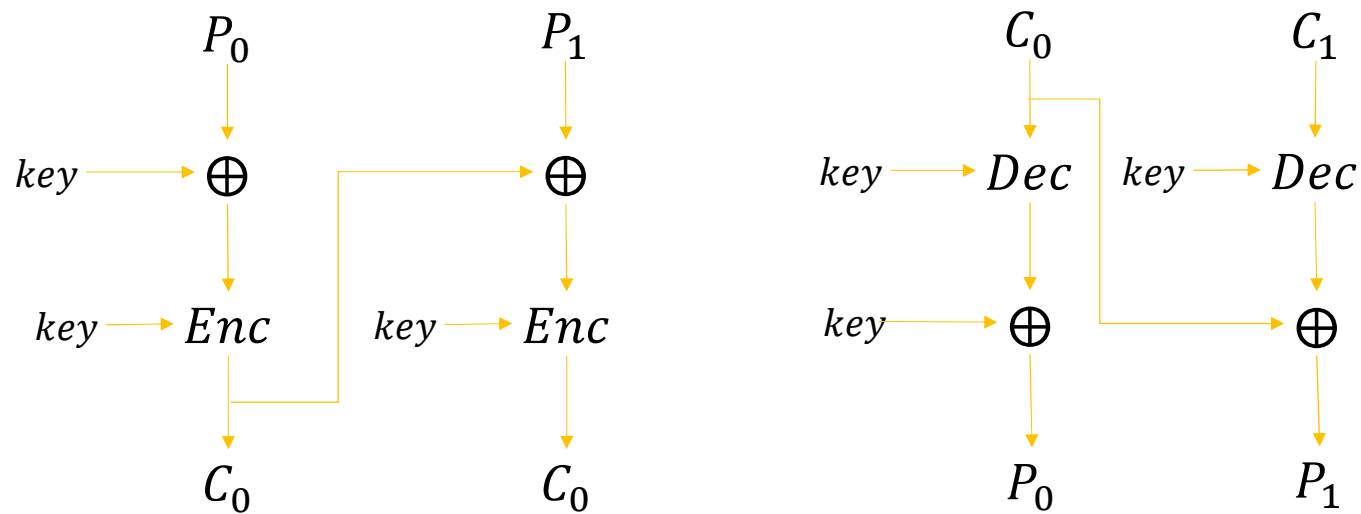
- Dreamhack:textbook-CBC

```
4
5 BLOCK_SIZE = 16
6 flag = open("flag", "rb").read()
7 key = bytes(randint(0, 255) for i in range(BLOCK_SIZE))
8
9 encrypt = lambda pt: AES.new(key, AES.MODE_CBC, key).encrypt(pad(pt, BLOCK_SIZE))
10 decrypt = lambda ct: unpad(AES.new(key, AES.MODE_CBC, key).decrypt(ct), BLOCK_SIZE)
11
12 print("Welcome to dream's AES server")
```

```
12 print("Welcome to dream's AES server")
13 while True:
14     print("[1] Encrypt")
15     print("[2] Decrypt")
16     print("[3] Get Flag")
17
18     choice = input()
19
20     if choice == "1":
21         print("Input plaintext (hex): ", end="")
22         pt = bytes.fromhex(input())
23         print(encrypt(pt).hex())
24
25     elif choice == "2":
26         print("Input ciphertext (hex): ", end="")
27         ct = bytes.fromhex(input())
28         print(decrypt(ct).hex())
29
30     elif choice == "3":
31         print(f"flag = {encrypt(flag).hex()}")
32         exit()
33
34     else:
35         print("Nope")
```

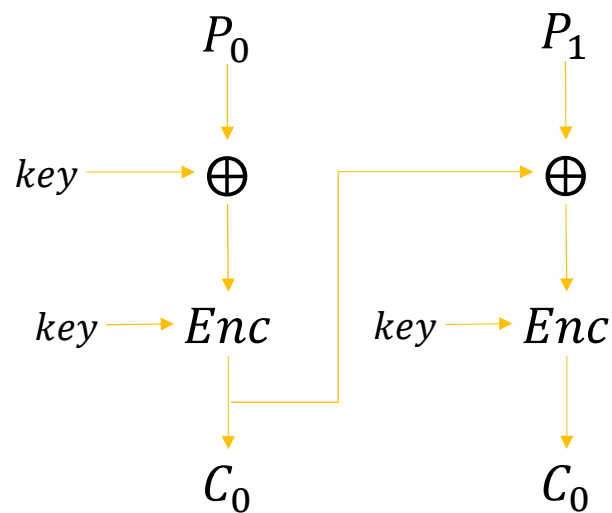
## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC



## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC



$P_0 = 0x00000000000000000000000000000000$  일때,

$$P_0 \oplus key = key$$

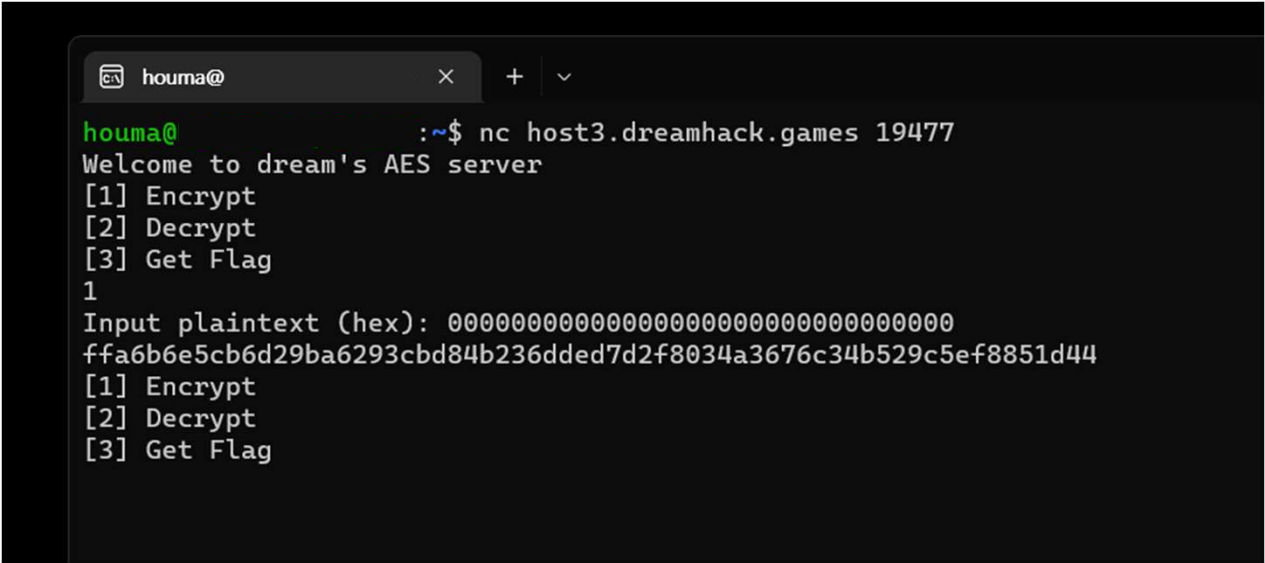
$$C_0 = Enc(P_0 \oplus key, key)$$

$$C_0 = Enc(key, key)$$

즉,  $C_0$ 는  $key$ 를 암호화한 값이 된다.

## 2.2 CBC Bit-Flipping Attack

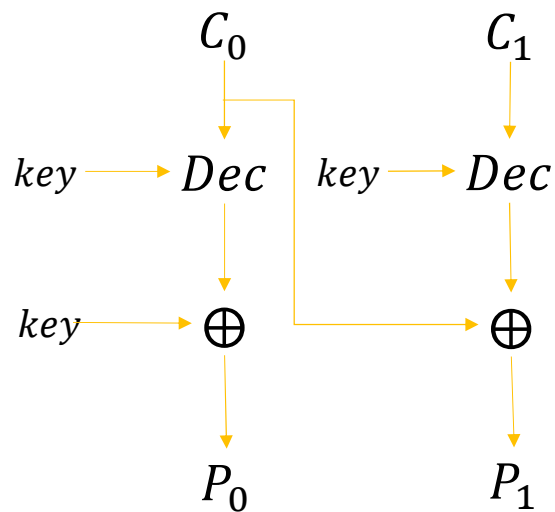
- Dreamhack:textbook-CBC



```
houma@ x + v
houma@ :~$ nc host3.dreamhack.games 19477
Welcome to dream's AES server
[1] Encrypt
[2] Decrypt
[3] Get Flag
1
Input plaintext (hex): 00000000000000000000000000000000
ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
[1] Encrypt
[2] Decrypt
[3] Get Flag
```

## 2.2 CBC Bit-Flipping Attack

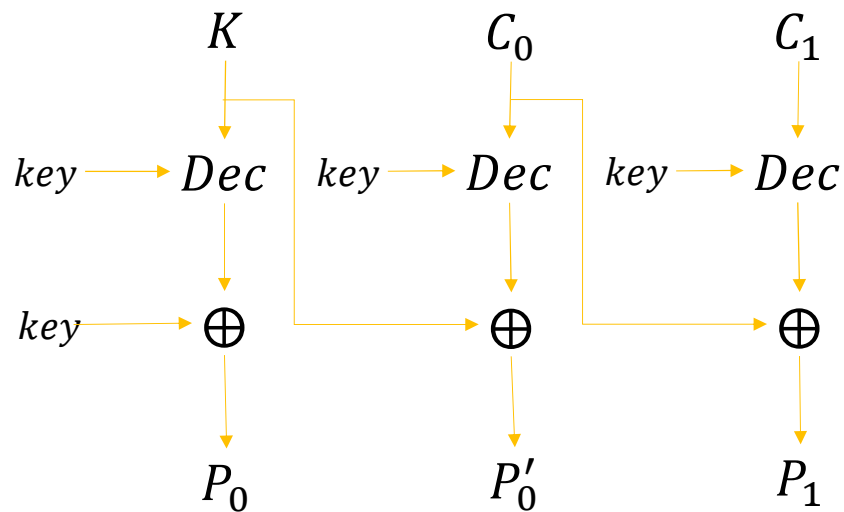
- Dreamhack:textbook-CBC



$C_0$ 는  $key$ 값을 암호화한 값이고,  
 $C_1$ 은 Padding을 암호화한 값이다.

## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC



암호문  $C_0||C_1$  앞에  
 $K = 0x000000000000000000000000000000000000$ 를 붙여  
 $K||C_0||C_1$ 을 들어 복호화를 한다.

$C_0$ 는  $key$ 를 암호화한 것으로, 복호화 한 후  
 $K$ 와 xor 연산을 하면  $P'_0$ 은  $key$ 값을 얻을 수 있다.

## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC

```
houma@  x + v
houma@ :~$ nc host3.dreamhack.games 19477
Welcome to dream's AES server
[1] Encrypt
[2] Decrypt
[3] Get Flag
1
Input plaintext (hex): 00000000000000000000000000000000
ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
[1] Encrypt
[2] Decrypt
[3] Get Flag
2
Input ciphertext (hex): 0000000000000000000000000000ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
ee18e08a4c163762314081d02659b20f1389ed8359d35a7653e5928b8e5a13d7
[1] Encrypt
[2] Decrypt
[3] Get Flag
```

## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC

```
houma@ ~$ nc host3.dreamhack.games 19477
Welcome to dream's AES server
[1] Encrypt
[2] Decrypt
[3] Get Flag
1
Input plaintext (hex): 00000000000000000000000000000000
ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
[1] Encrypt
[2] Decrypt
[3] Get Flag
2
Input ciphertext (hex): 00000000000000000000000000000000ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
ee18e08a4c163762314081d02659b20-1389ed8359d35a7653e5928b8e5a13d7
[1] Encrypt
[2] Decrypt
[3] Get Flag
```

$P'_0$ 부분이 *key*값임을 알 수 있다.



## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC

```
houma@ ~$ nc host3.dreamhack.games 19477
Welcome to dream's AES server
[1] Encrypt
[2] Decrypt
[3] Get Flag
1
Input plaintext (hex): 00000000000000000000000000000000
ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
[1] Encrypt
[2] Decrypt
[3] Get Flag
2
Input ciphertext (hex): 0000000000000000000000000000ffa6b6e5cb6d29ba6293cbd84b236dded7d2f8034a3676c34b529c5ef8851d44
ee18e08a4c163762314081d02659b20f1389ed8359d35a7653e5928b8e5a13d7
[1] Encrypt
[2] Decrypt
[3] Get Flag
3
flag = 1028969502214ab923679cc66720bfbf70d0491aeb2bcfb3a0a8d9087f0f14bf7a039346f627d80d51876a1fd48feaa0
```

## 2.2 CBC Bit-Flipping Attack

- Dreamhack:textbook-CBC

```
1  from Crypto.Util.Padding import pad, unpad
2  from Crypto.Cipher import AES
3  from pwn import *
4
5  encrypt = lambda pt: AES.new(key, AES.MODE_CBC, key).encrypt(pad(pt, 16))
6  decrypt = lambda ct: unpad(AES.new(key, AES.MODE_CBC, key).decrypt(ct), 16)
7
8  key = bytes.fromhex('ee18e08a4c163762314081d02659b20f1389ed8359d35a7653e5928b8e5a13d7')[16:32]
9  flagct = bytes.fromhex('1028969502214ab923679cc66720bfbf70d0491aeb2bcfb3a0a8d9087f0f14bf7a039346f627d80d51876a1fd48feaa0')
10
11 print(decrypt(flagct))
12
```

문제 7 출력 디버그 콘솔 터미널 JUPYTER

+ v bash

```
(base)
houma@ MINGW64 /
$ python test.py
b'DH{9666eb07031fbc855428e2223946a4b8}\n'
```

감사합니다.