CS 370: Introduction to Security

04.18: Block-cipher and symmetric enc. (cont'd)

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab

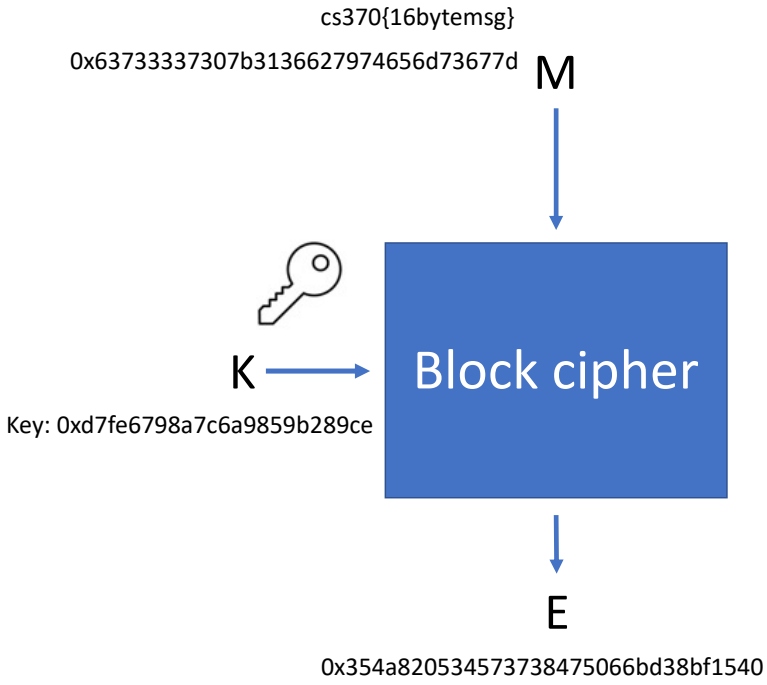
Topics for today

- Recap
  - Block ciphers
  - Block cipher modes
  - ECB weaknesses
- Block cipher modes
  - How can an adversary exploit the ECB's weakness (Micro-labs)?
  - How can we address the ECB's weakness?
  - How secure is CBC and can an adversary exploit it (Micro-labs)?
  - How can we address the CBC's weakness?

Block cipher: encryption

- Block cipher
  - Cryptographic algorithm that work only with fixed-length set of bits

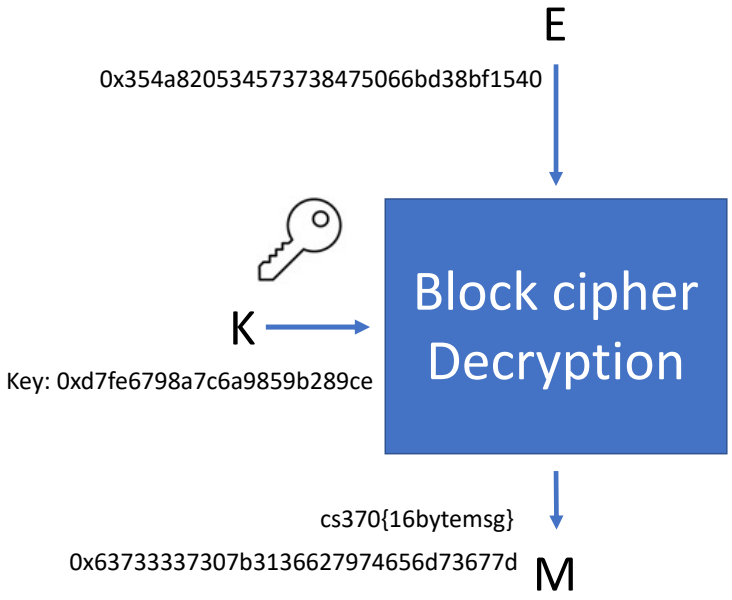
- Terminology
  - **Block:** a fixed size message *M*
  - **Key:** a secret we use for encryption
    - Shared between a sender and a receiver
  - **Encryption:** use *K* to convert *M* into *E*



Block cipher: decryption

- Block cipher
  - Cryptographic algorithm that work only with fixed-length set of bits

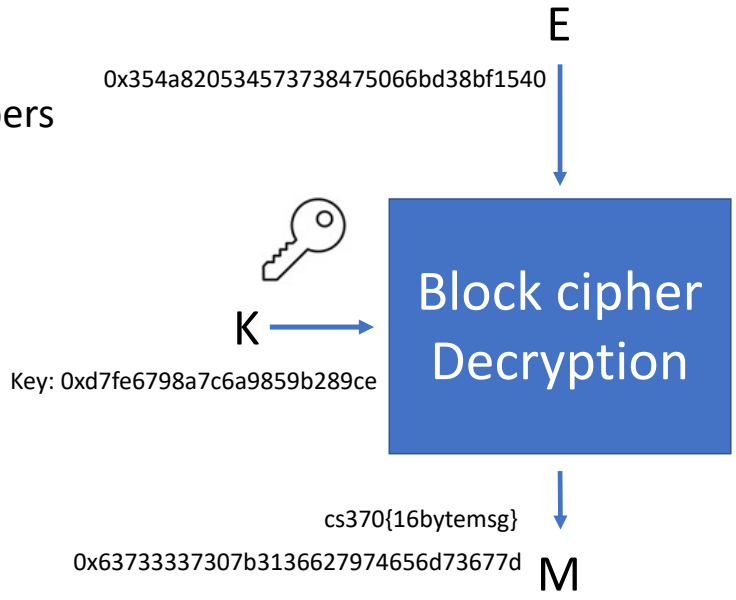
- Terminology
  - **Block:** a fixed size message *M*
  - **Key:** a secret we use for encryption
    - Shared between a sender and a receiver
  - **Encryption:** use *K* to convert *M* into *E*
  - **Decryption:** use *K* to convert *E* into *M*



Block cipher

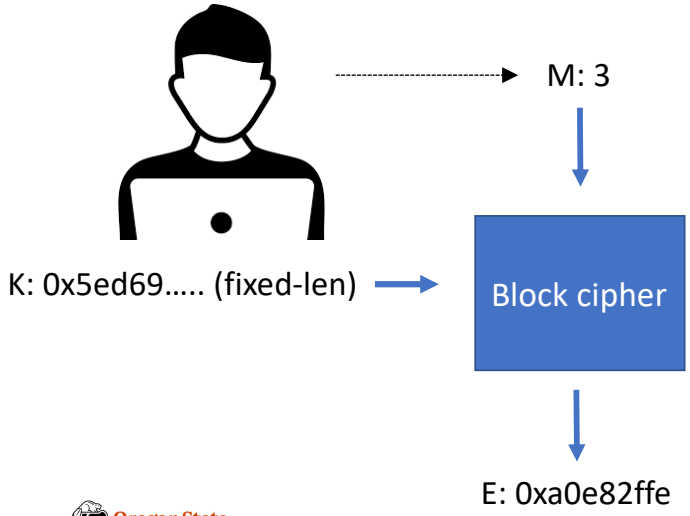
- Formally
  - You can see encryption and decryption as
  - Generating a permutation of numbers:
    - $\{0,1\}^n$ -> $\{0,1\}^n$
    - Mappings should be 1-to-1
  - The key determines how to permute the numbers

M	Ciphertext
0	0xaf531b0e1
1	0x14a986e7a
2	0xad738009d
3	0x5ed6985c5
4	0xf3b8aa2e8
5	0xad04ec00e
…	0x59fd94c21



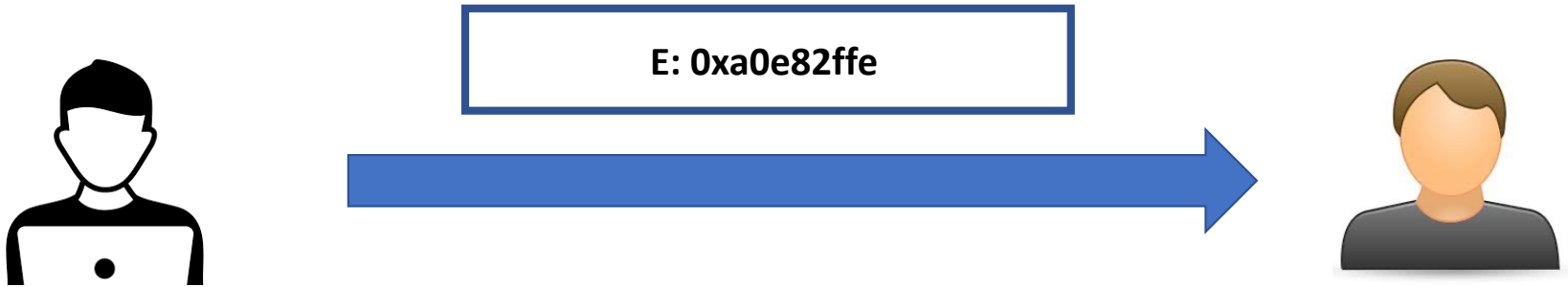
Block cipher: in operation

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm
    - 1st: Share the information about the key to use



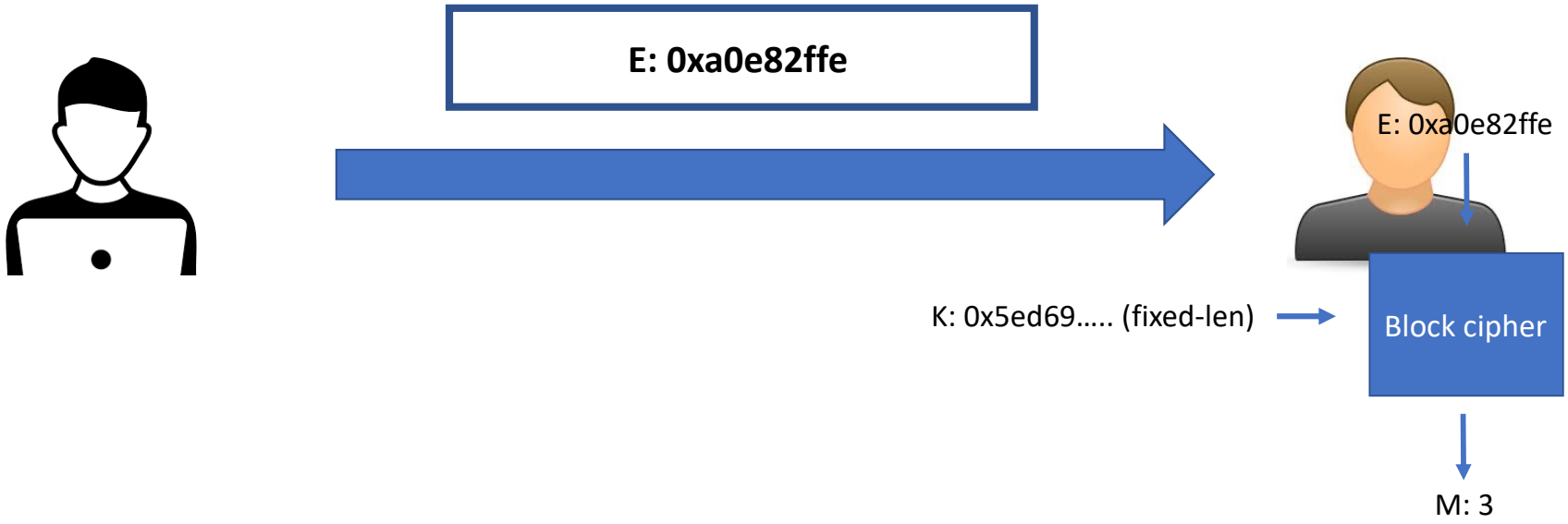
Block cipher: in operation

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm
    - 1st: Share the information about the key to use



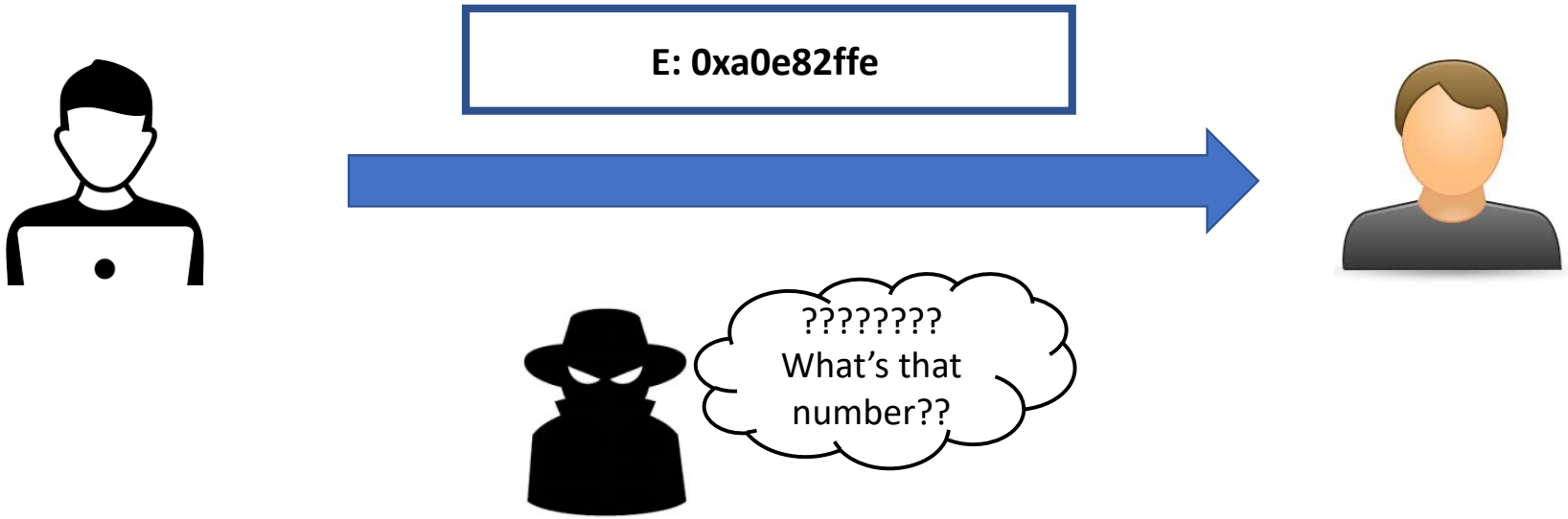
Block cipher: in operation

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm
    - 1st: Share the information about the key to use



Block cipher: in operation

- Goal
  - We want to communicate with others securely (and privately)
  - Both parties use the same block cipher algorithm
    - 1st: Share the information about the key to use



Electronic code block

- ECB
  - A mode of block cipher operations
  - We pad the length of a message at the end

- ECB Operation
  - Suppose that we encrypt 15-byte data: 0123456789ABCDE (e.g., 0 = 0x30)
  - ECB pads 0x01 (= 1-byte length) at the end

Pos	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
Hex	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x41	0x42	0x43	0x44	0x45	0x01

Electronic code block – cont'd

- ECB
  - A mode of block cipher operations
  - We pad the length of a message at the end

- ECB Operation (corner-case)
  - Suppose that we encrypt 16-byte data: 0123456789ABCDE\x01 (e.g., 0 = 0x30)
  - How we can distinguish this from 15-byte data with 0x01 padding
  - We pad 16-byte of 0x10 at the end (= we encrypt two blocks)

Pos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x41	0x42	0x43	0x44	0x45	0x01

Pos	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hex	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10	0x10

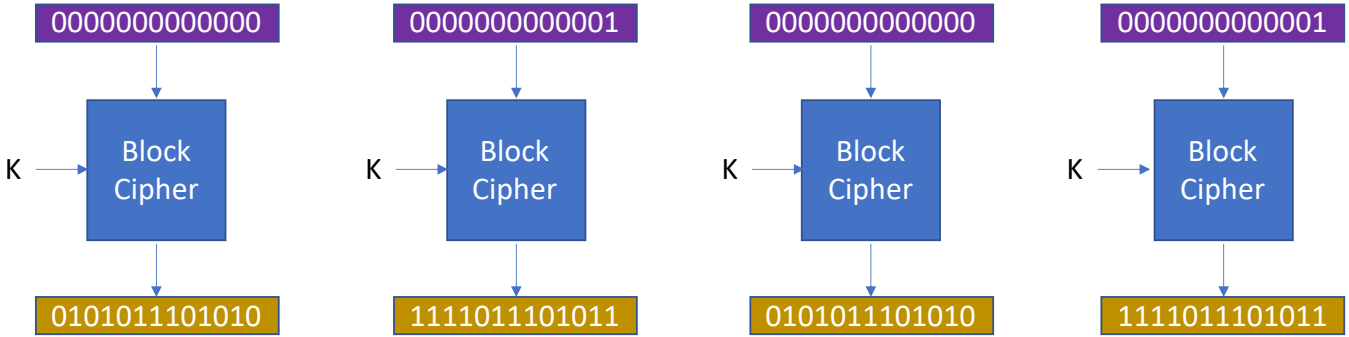
Electronic code block – cont'd

- ECB
  - A mode of block cipher operations
  - We pad the length of a message at the end

- ECB Operation (corner-case)
  - Suppose that we encrypt 31-byte data: 0123456789ABCDEF0123456789ABCDE
  - How can we encrypt/decrypt this message?
    - Split the message into 16-bytes: 0123456789ABCDEF + 0123456789ABCDE
    - Encrypt the first block: 0123456789ABCDEF
    - Encrypt the second block (with pads): 0123456789ABCDE\x01
  - You can encrypt each block in parallel

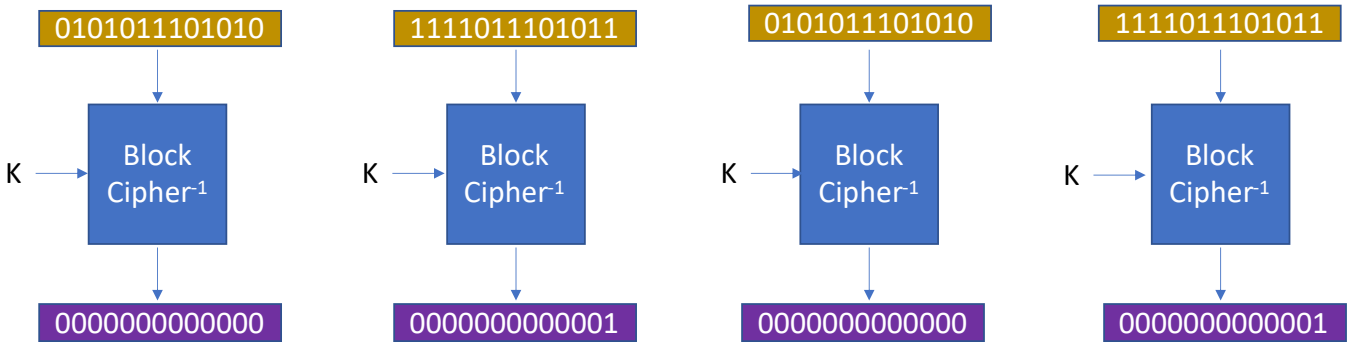
Electronic code block – cont'd

- ECB Operation
  - You can encrypt each block in parallel



Electronic code block – cont'd

- ECB Operation
  - You can encrypt (and decrypt) each block in parallel

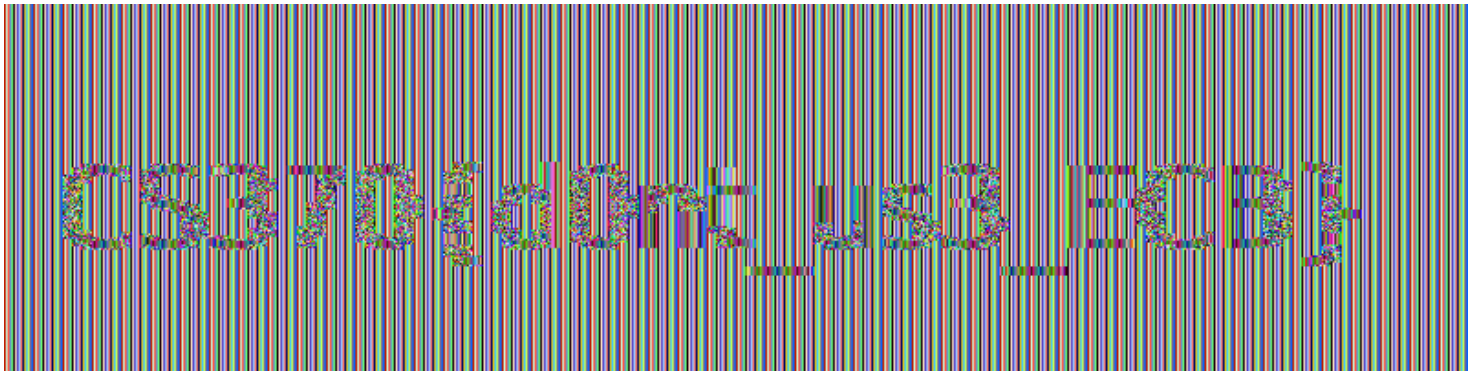


Electronic code block – cont'd

- ECB weakness(es)
  - Using the same key leads to the same ciphertext
  - An adversary can guess the message by looking at the ciphertext
  - Suppose:
    - M: 0 -> C: 0x39827332…
    - M: 1 -> C: 0x5a83f874…
    - …

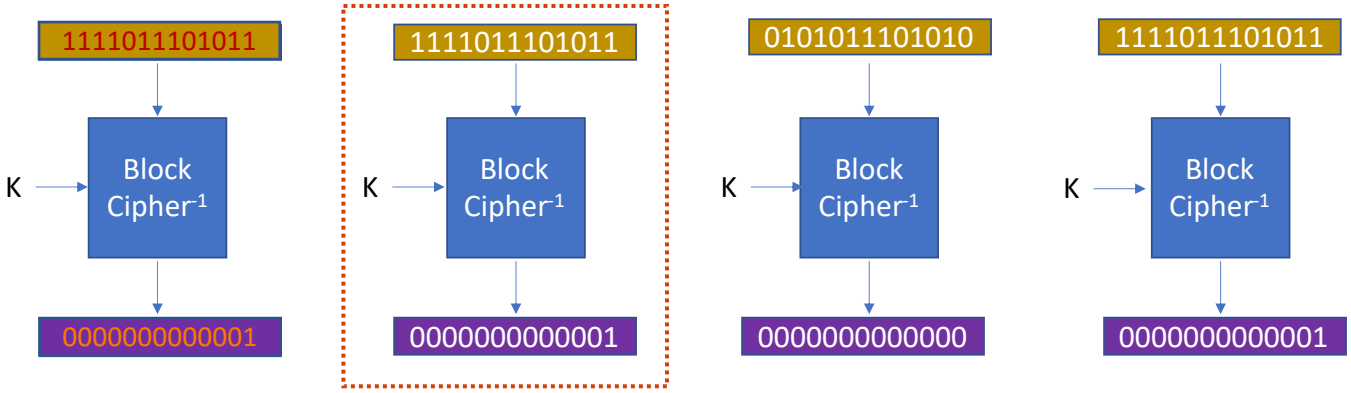
Micro-labs

- ECB weakness
  - I will provide you a super-secretly-encrypted photo
  - Your job is to guess what's in the photo



Electronic code block – cont'd

- ECB weakness(es)
  - Using the same key leads to the same ciphertext
  - An adversary can guess the message by looking at the ciphertext
  - An adversary can modify the ciphertext to compromise the plaintext



Attacker knows this

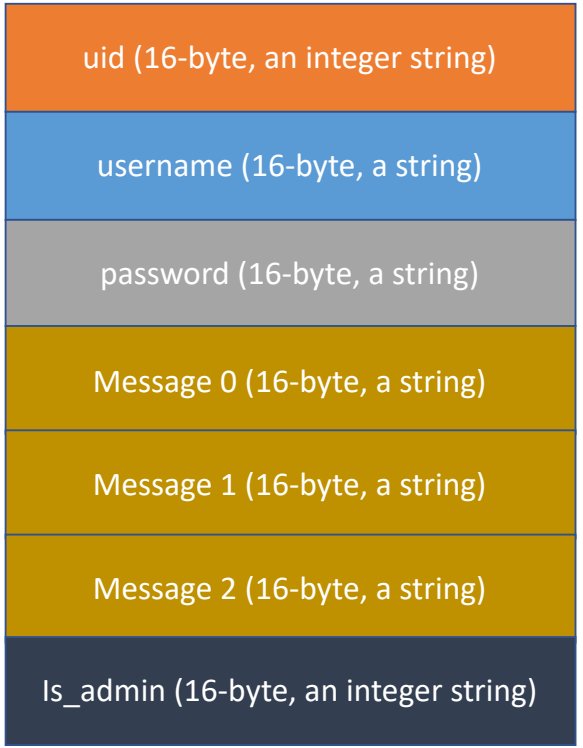
Topics for today

- Recap
  - Block ciphers
  - Block cipher modes
  - ECB weaknesses
- Block cipher modes
  - How can an adversary exploit the ECB's weakness (Micro-labs)?
  - How can we address the ECB's weakness?
  - How secure is CBC and can an adversary exploit it (Micro-labs)?
  - How can we address the CBC's weakness?

Micro-labs: how can an adversary exploit the ECB's weakness?

- Goals
  - Get the three flags by exploiting the ECB weakness
- Starter
  - Go to ~/week2/ecb-attack and run ./launcher
  - Enter encrypted.user

```
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ls
config  ecb-decryptor.py  ecb-encryptor.py  encrypted.user  flags  install.py  key  launcher  output.txt  template.py  user.py
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week2/ecb-attack/ecb-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: encrypted.user
Decrypted user information:
       uid      : 1
  username      : 'notadministrator'
  password      : 'passwordpassword'
   message      : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin      : 0

Raw data: '0000000000000001notadministratorpasswordpasswordThe sky is blue, cs370 crypto is a boring class.0000000000000000'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin == 1
3. I changed the password to something else
4. quit
Your choice (1-4):
```



Micro-labs: how can an adversary exploit the ECB's weakness?

- Goals
  - Get the three flags by exploiting the ECB weakness
- Starter
  - Go to ~/week2/ecb-attack and run ./launcher
  - Enter encrypted.user

```
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ls
config  ecb-decryptor.py  ecb-encryptor.py  encrypted.user  flags  install.py  key  launcher  output.txt  template.py  user.py
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week2/ecb-attack/ecb-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: encrypted.user
Decrypted user information:
       uid      : 1
  username      : 'notadministrator'
  password      : 'passwordpassword'
   message      : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin      : 0

Raw data: '0000000000000001notadministratorpasswordpasswordThe sky is blue, cs370 crypto is a boring class.0000000000000000'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin == 1
3. I changed the password to something else
4. quit
Your choice (1-4):
```

Micro-labs: how can an adversary exploit the ECB's weakness?

- Job 1
  - Create a copy of this data with 'uid == 0'
  - Use template.py (marked as XXX)

- Hint
  - Find the ciphertext corresponding to 0

```
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ls
config  ecb-decryptor.py  ecb-encryptor.py  encrypted.user  flag1.user  flag2.user  flag3.user  flags  install.py  key  launcher
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week2/ecb-attack/ecb-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: flag1.user
Decrypted user information:
       uid      : 0
  username      : 'notadministrator'
  password      : 'passwordpassword'
   message      : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin      : 0

Raw data: '0000000000000000notadministratorpasswordpasswordThe sky is blue, cs370 crypto is a boring class.0000000000000000'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin == 1
3. I changed the password to something else
4. quit
Your choice (1-4): 1
```

Micro-labs: how can an adversary exploit the ECB's weakness?

- Job 2
  - Create a copy of this data with 'is_admin == 1'
  - Use template.py (marked as XXX)

- Hint
  - Find the ciphertext corresponding to 1

```
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ls
config  ecb-decryptor.py  ecb-encryptor.py  encrypted.user  flag1.user  flag2.user  flag3.user  flags  install.py  key  launcher
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week2/ecb-attack/ecb-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: flag2.user
Decrypted user information:
       uid      : 1
  username      : 'notadministrator'
  password      : 'passwordpassword'
   message      : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin      : 1

Raw data: '0000000000000001notadministratorpasswordpasswordThe sky is blue, cs370 crypto is a boring class.0000000000000001'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin == 1
3. I changed the password to something else
4. quit
Your choice (1-4): 2
```

Micro-labs: how can an adversary exploit the ECB's weakness?

- Job 3
  - Create a copy of this data with a different password
  - Use template.py (marked as XXX)

- Hint
  - Find the ciphertext not corresponding to 'passwo…'

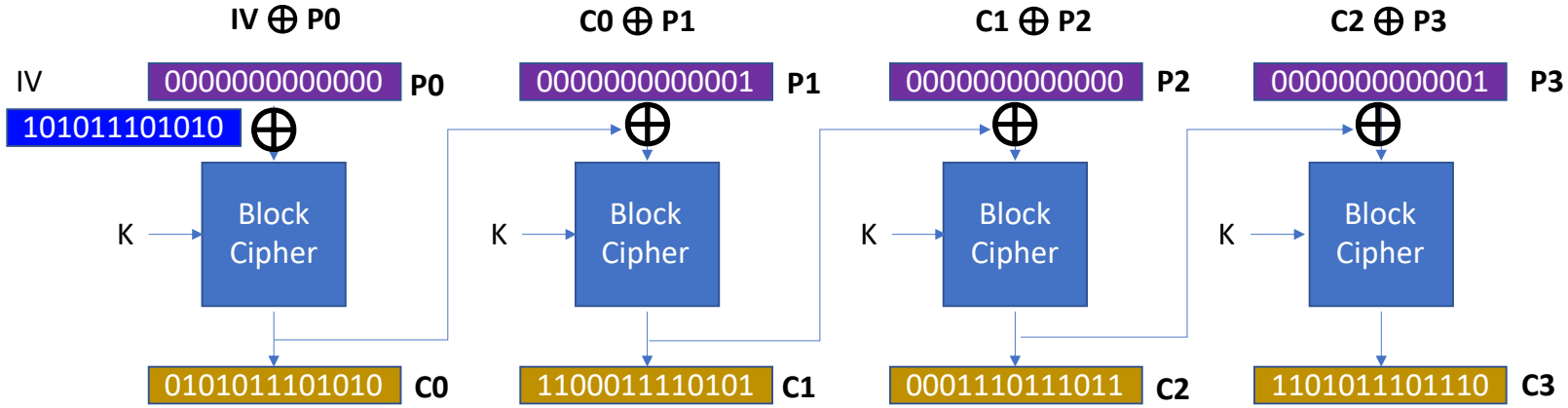
```
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ls
config  ecb-decryptor.py  ecb-encryptor.py  encrypted.user  flag1.user  flag2.user  flag3.user  flags  install.py  key  launcher
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$
neuronoverflow@ip-172-31-3-119:~/week2/ecb-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week2/ecb-attack/ecb-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: flag3.user
Decrypted user information:
      uid      : 1
  username     : 'notadministrator'
  password     : 'a boring class.'
   message     : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin     : 0

Raw data: '0000000000000001notadministrator a boring class.The sky is blue, cs370 crypto is a boring class.0000000000000000'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin == 1
3. I changed the password to something else
4. quit
Your choice (1-4): 3
```

Topics for today

- Recap
  - Block ciphers
  - Block cipher modes
  - ECB weaknesses
- Block cipher modes
  - Exploiting the ECB's weakness (Micro-labs)?
  - How can we address the ECB's weakness?
  - How secure is CBC and can an adversary exploit it (Micro-labs)?
  - How can we address the CBC's weakness?

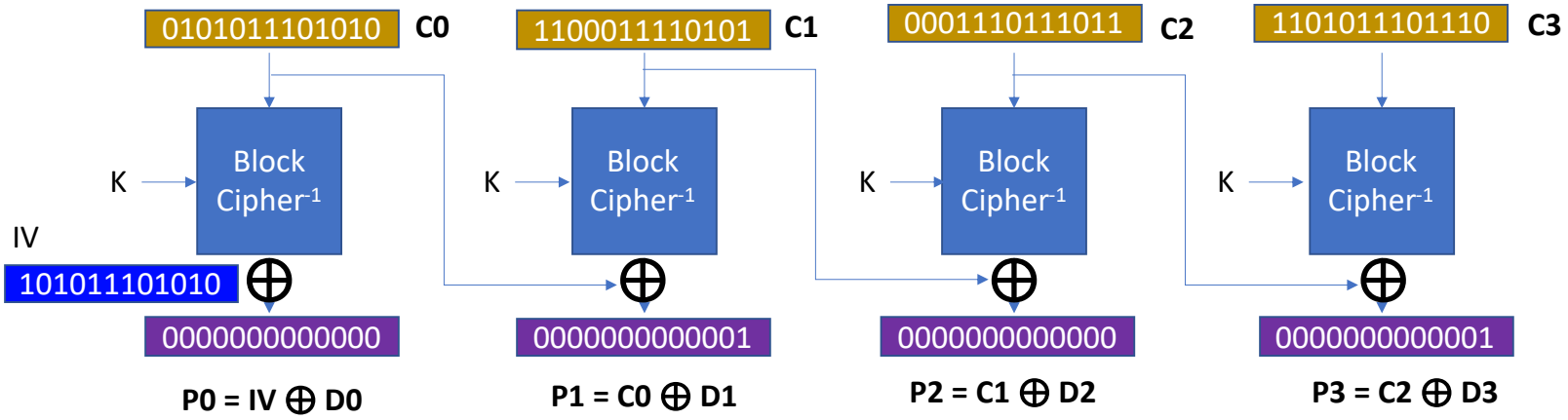
Cipher block chain

- CBC
  - A mode of block cipher operations
  - Operations
    - M: XOR between IV (initialization vector) and the P0 (plaintext)
    - Encryption: use the ciphertext from the prev. block as IV and run block encryption



Cipher block chain – cont'd

- CBC
  - A mode of block cipher operations
  - Operations
    - M: XOR between IV (initialization vector) and the P0 (plaintext)
    - Encryption: use the ciphertext from the prev. block as IV and run block encryption
    - Decryption: user the ciphertext from the prev. block as IV and run block decryption



Cipher block chain – cont'd

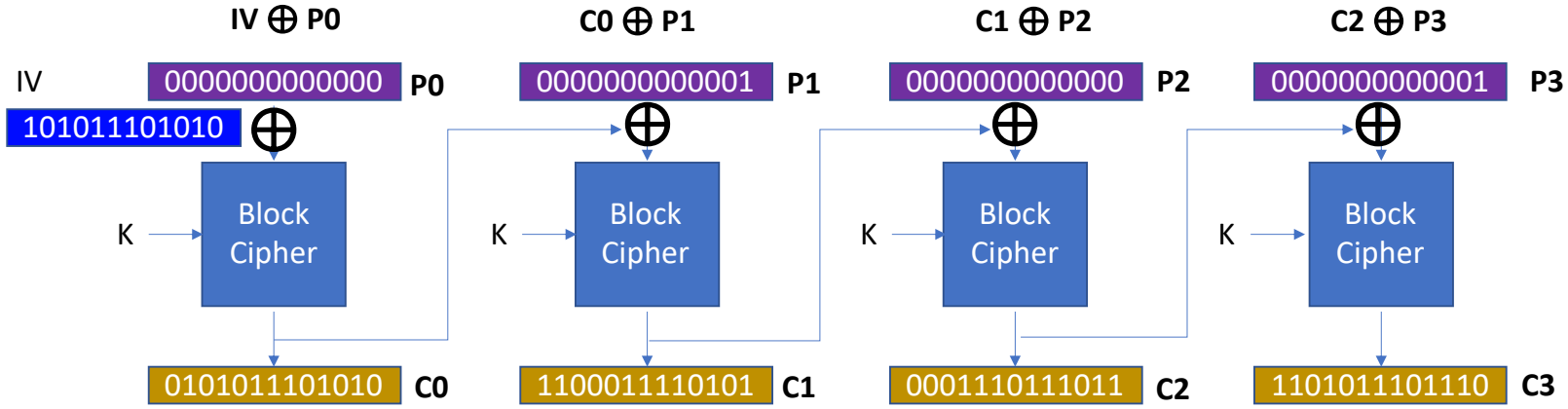
- CBC
  - A mode of block cipher operations
  - Operations
    - M: XOR between IV (initialization vector) and the P0 (plaintext)
    - Encryption: use the ciphertext from the prev. block as IV and run block encryption
    - Decryption: user the plaintext from the prev. block as IV and run block decryption
  - Benefits
    - Address the ECB's weakness
      - Both encryption and decryption are not deterministic
      - We can do this by using a random IV
    - Check it out by yourself: link to cbc-encrypted image

Topics for today

- Recap
  - Block ciphers
  - Block cipher modes
  - ECB weaknesses
- Block cipher modes
  - Exploiting the ECB's weakness (Micro-labs)?
  - Cipher block chain (CBC)
  - How secure is CBC and can an adversary exploit it (Micro-labs)?
  - How can we address the CBC's weakness?

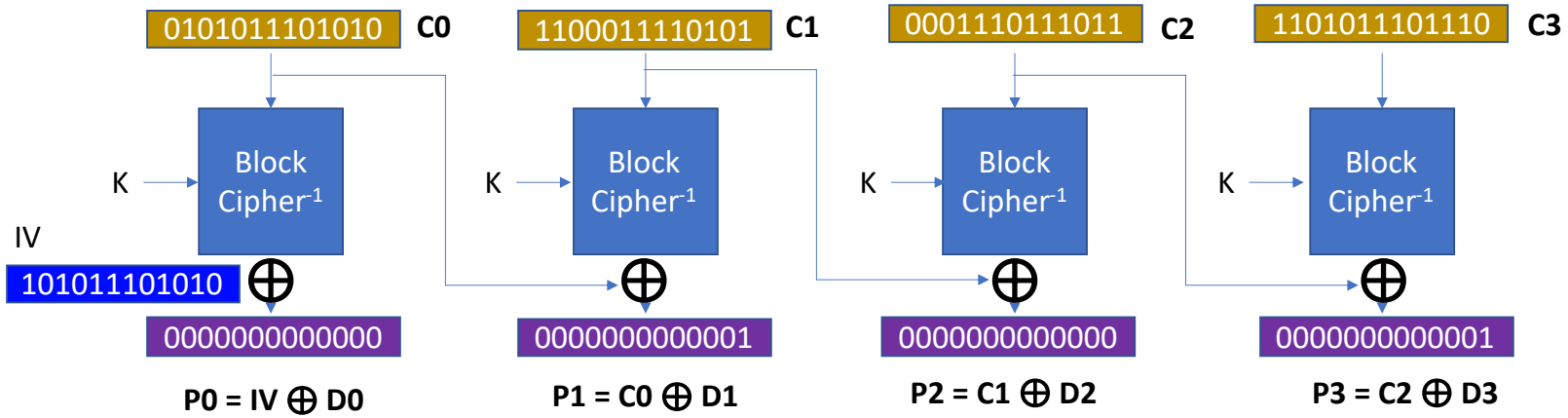
Cipher block chain

- CBC weakness
  - Can't run encryption in parallel



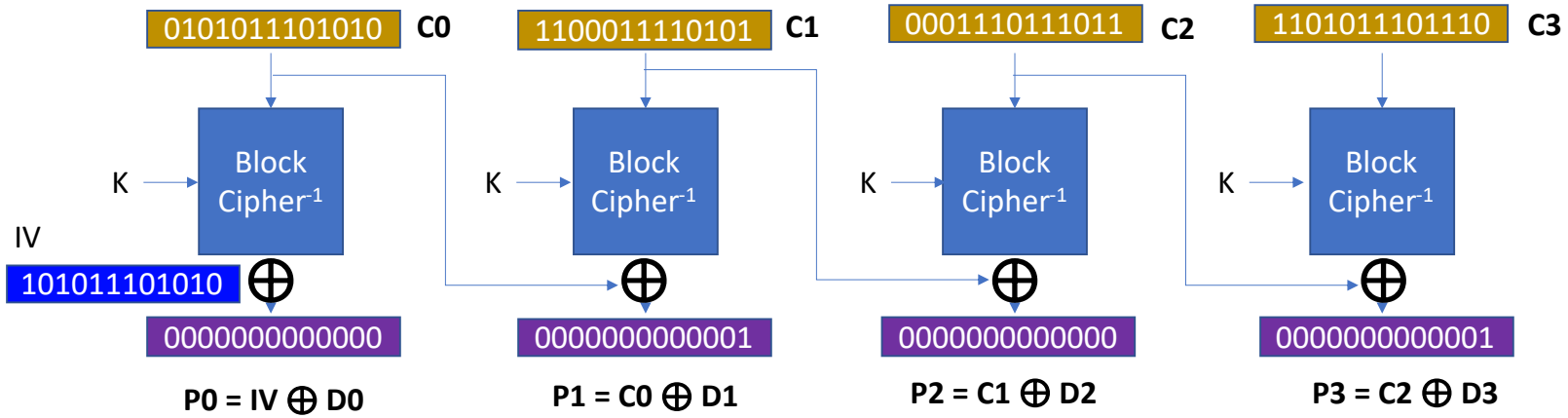
Cipher block chain

- CBC weakness
  - Can't run encryption in parallel
  - But can run decryption in parallel (why this is a weakness?)



Cipher block chain

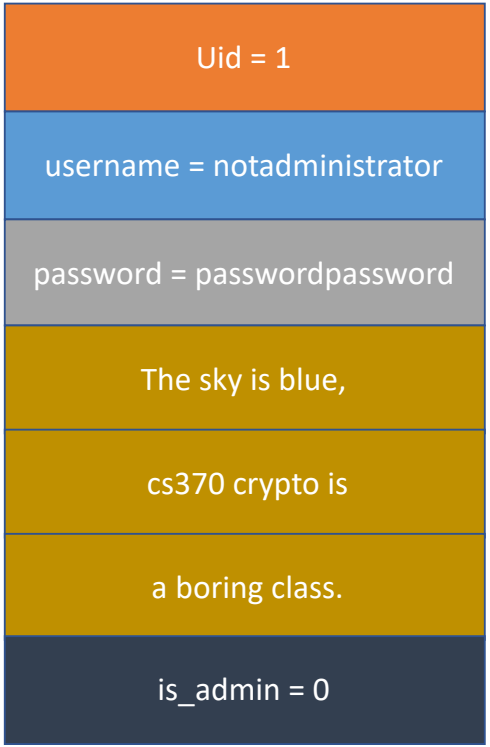
- CBC weakness
  - Can't run encryption in parallel
  - But can run decryption in parallel
  - We can infer the dependency in decryption



Micro-labs: how can an adversary exploit the CBC's weakness?

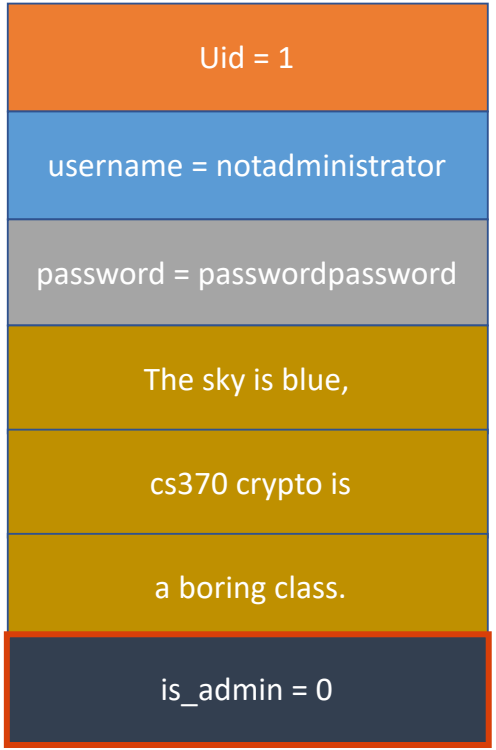
- Goals
  - Get the three flags by exploiting the CBC's weakness
- Starter
  - Go to ~/week3/cbc-attack and run ./launcher
  - Enter encrypted.user

```
neuronoverflow@ip-172-31-3-119:~/week3/cbc-attack$ ls
cbc-decryptor.py  cbc-encryptor.py  config  encrypted.user  flag1.user  flag2.user  flag3.user  flags  install.py  key  launcher
neuronoverflow@ip-172-31-3-119:~/week3/cbc-attack$ ./launcher
Running Command: [/root/Microlab-Problems/problems/week3/cbc-attack/cbc-decryptor.py]
Key was loaded successfully!
Give me the filename of your encrypted object: encrypted.user
Decrypted user information:
       uid      : 1
  username      : 'notadministrator'
  password      : 'passwordpassword'
   message      : 'The sky is blue, cs370 crypto is a boring class.'
  is_admin      : 0

Raw data: '0000000000000001notadministratorpasswordpasswordThe sky is blue, cs370 crypto is a boring class.0000000000000000'


Choose which flag do you want to get:
1. I made uid == 0 (super user)
2. I made is_admin != 0
3. I changed the message from:
a boring class.
to:
a superb class.
4. quit
Your choice (1-4):
```



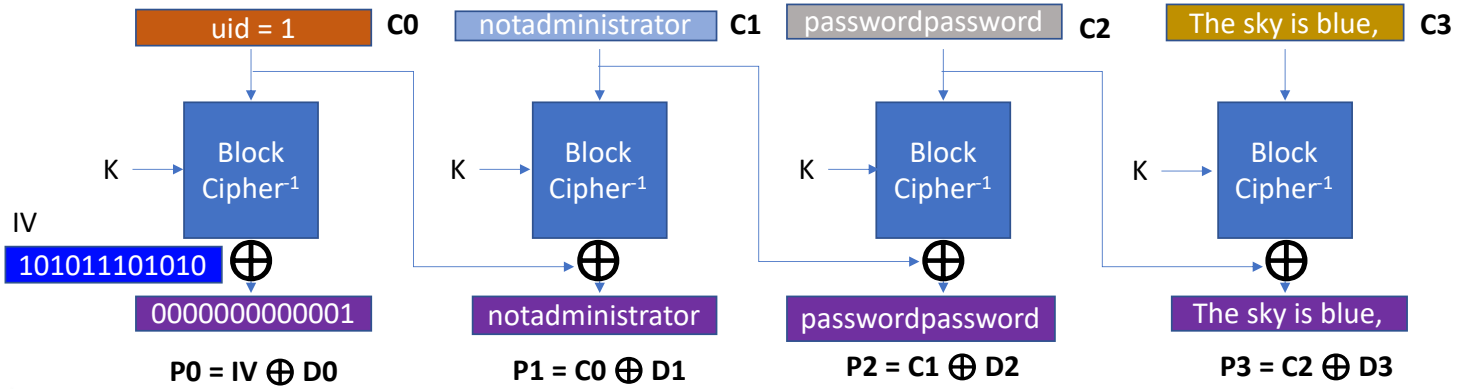
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 1
  - Create a copy of this data with 'uid == 0'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 1st block



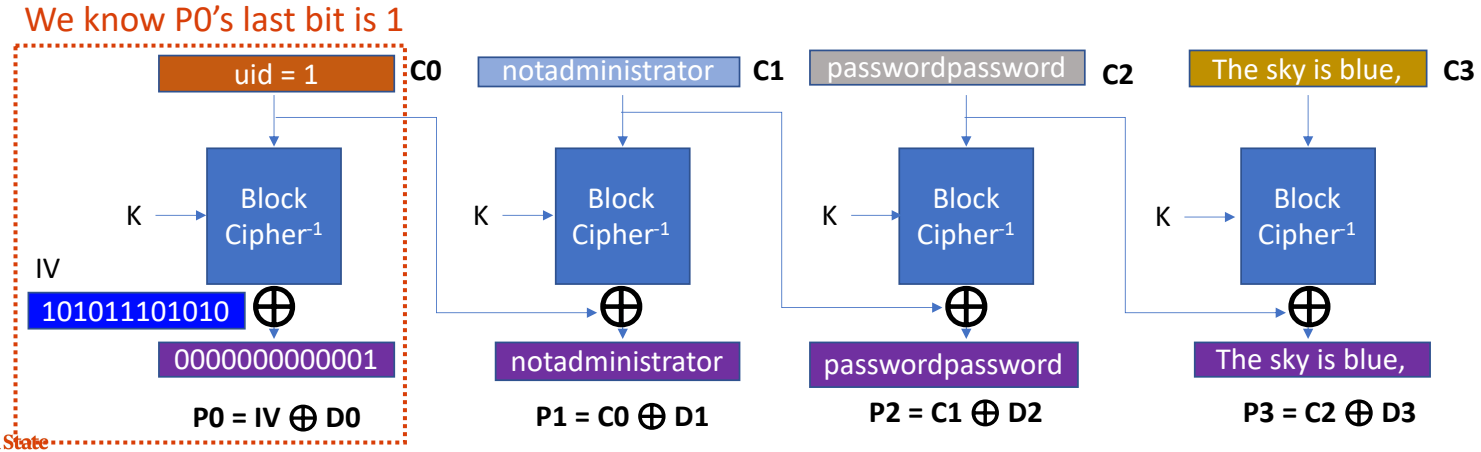
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 1
  - Create a copy of this data with 'uid == 0'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 1st block



Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 1
  - Create a copy of this data with 'uid == 0'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 1st block

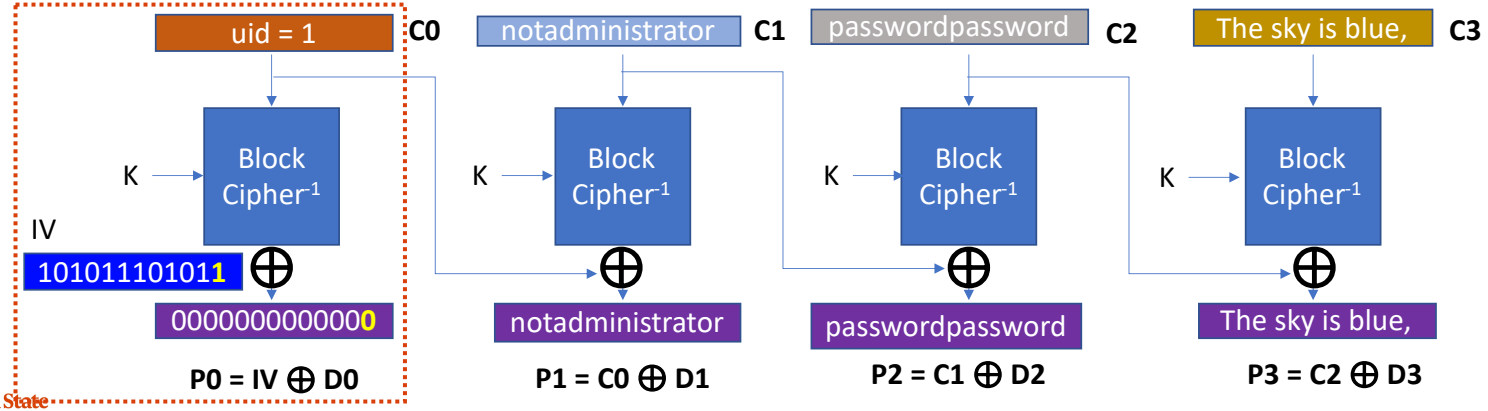


Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 1
  - Create a copy of this data with 'uid == 0'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 1st block

```python
def create_file_for_flag_1():
    bytestring = ''

    # use whatever copied array here
    copied_blocks_bytes = copy.deepcopy(blocks_bytes)
    copied_blocks_hex = copy.deepcopy(blocks_hex)
    copied_blocks_int = copy.deepcopy(blocks_int)

    # XXX: Your code here; transform the blocks here
    copied_blocks_int[0][-1] ^= 1

    # in case you used blocks_int
    bytestring = convert_int_blocks_to_bytestring(copied_blocks_int)

    # write as flag1.user
    with open("flag1.user", "wb") as f:
        f.write(bytestring)
```

What if we flip IV's last bit from 0 to 1

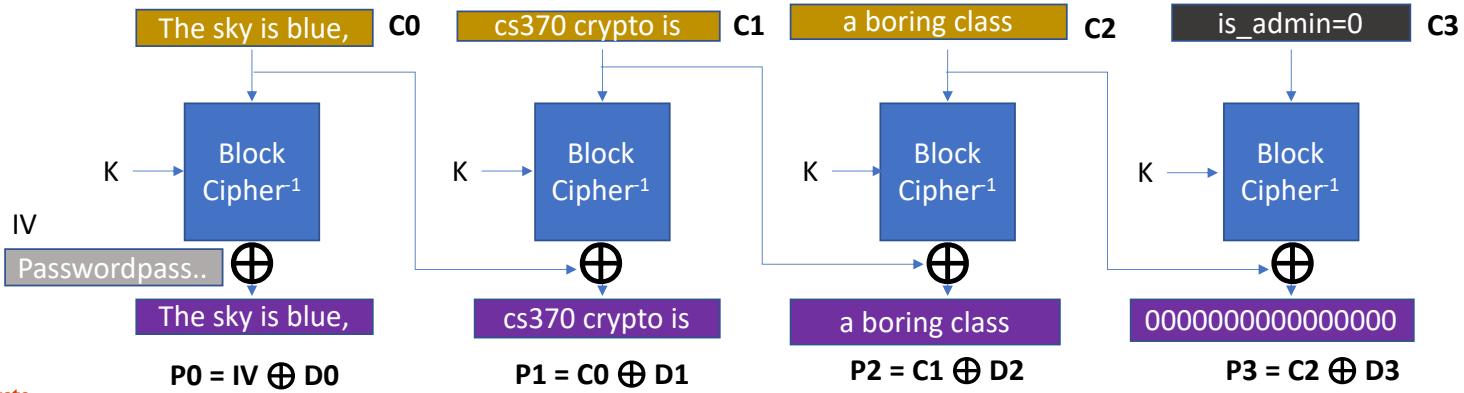


Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 2
  - Create a copy of this data with 'is_admin == 1'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 6th block

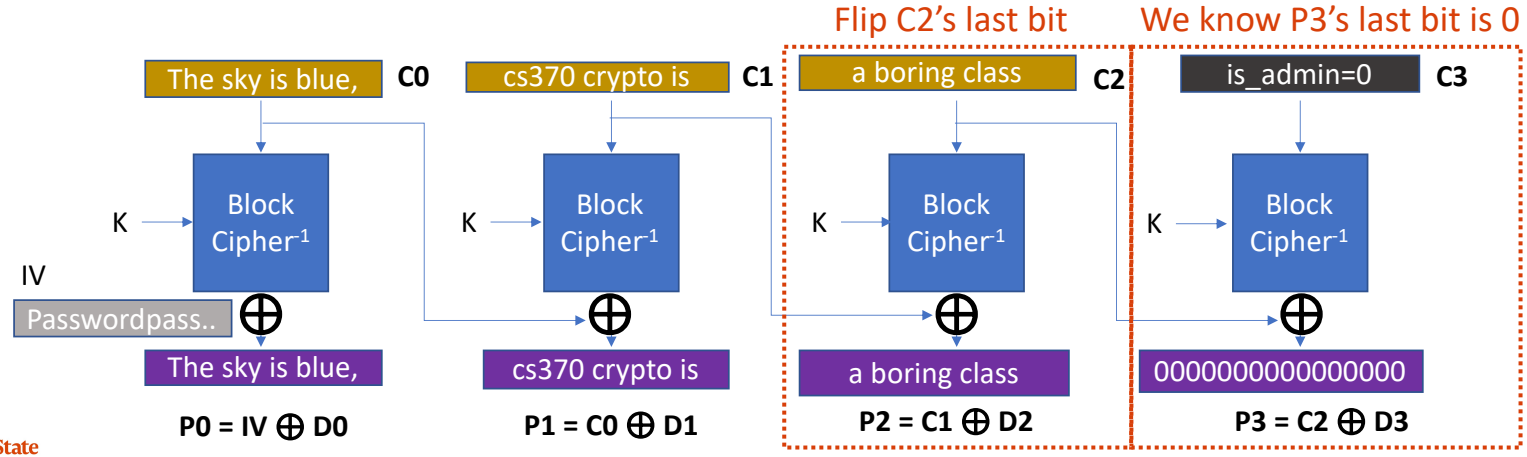
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 2
  - Create a copy of this data with 'is_admin == 1'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 6th block



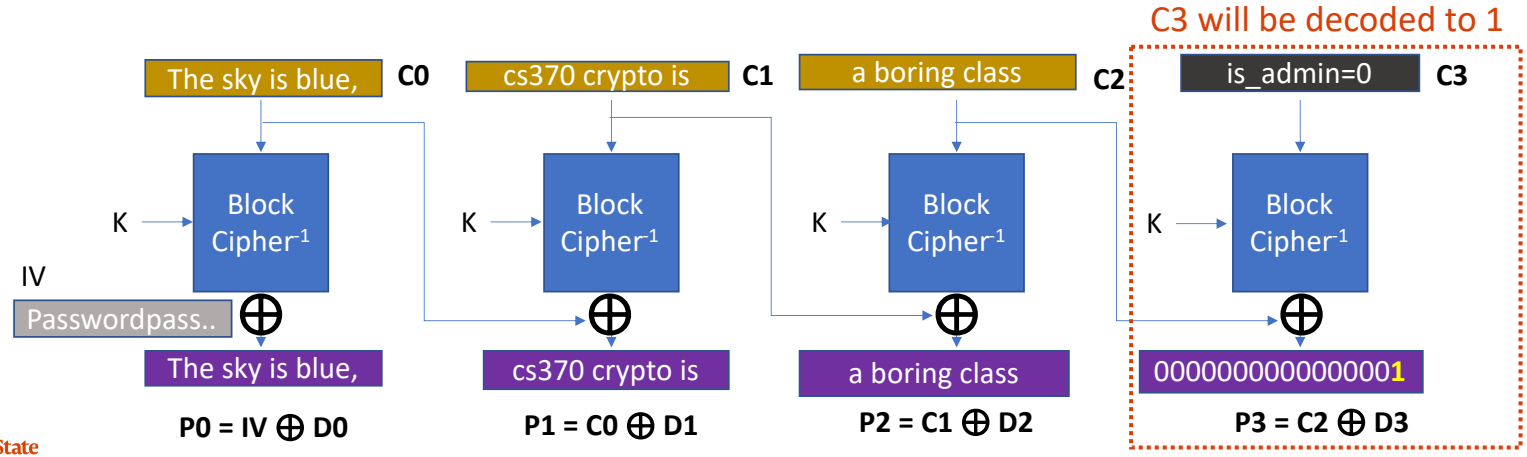
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 2
  - Create a copy of this data with 'is_admin == 1'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 6th block



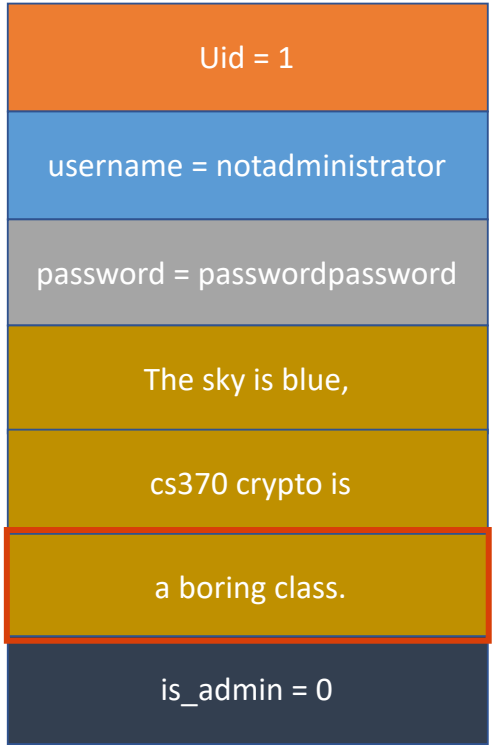
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 2
  - Create a copy of this data with 'is_admin == 1'
  - Use template.py (marked as XXX)
  - (Warning) we cannot use the last block
- Hint
  - Find a way to flip the decrypted value of the 6th block



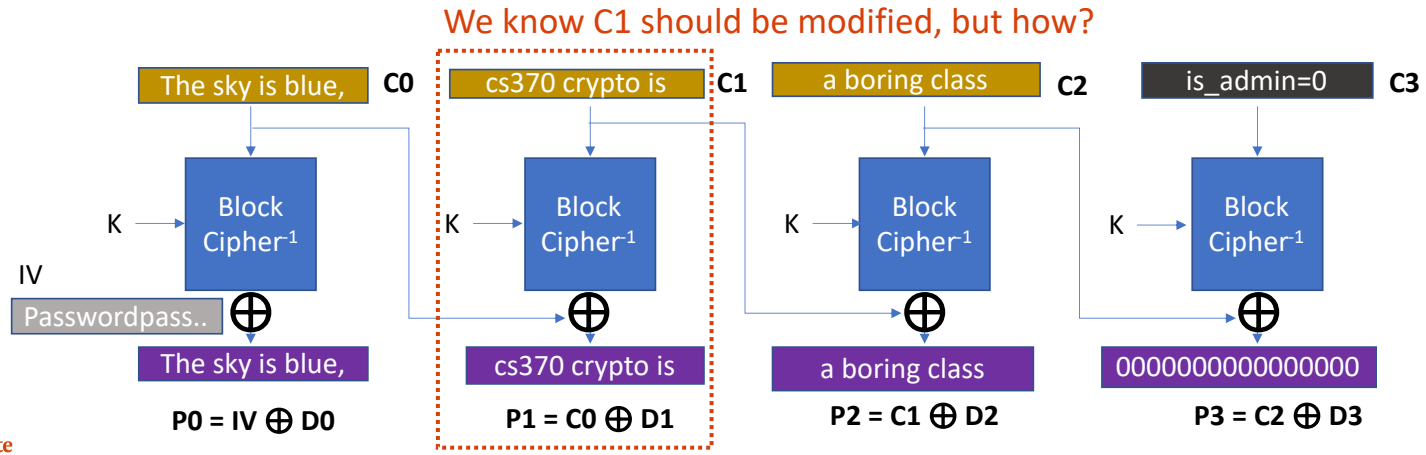
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 3
  - Create a copy of this data with
  - The change from 'boring' to 'superb'
  - Use template.py (marked as XXX)
- Hint
  - Find a way to modify the plaintext of the 5th block



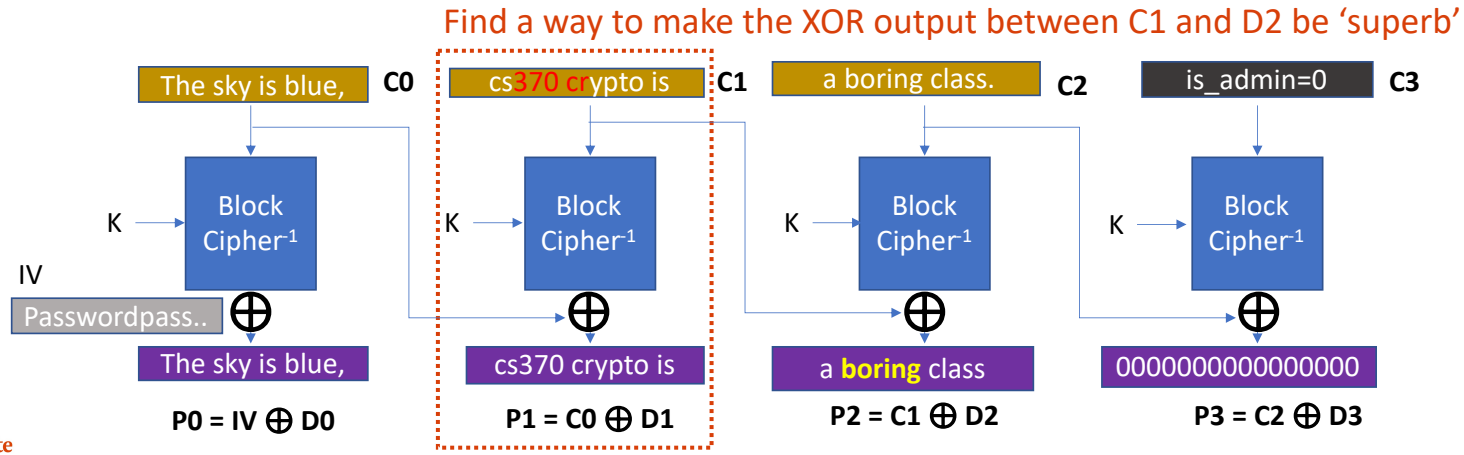
Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 3
  - Create a copy of this data with
  - The change from 'boring' to 'superb'
  - Use template.py (marked as XXX)
- Hint
  - Find a way to modify the plaintext of the 5th block



Micro-labs: how can an adversary exploit the CBC's weakness?

- Job 3
  - Create a copy of this data with
  - The change from 'boring' to 'superb'
  - Use template.py (marked as XXX)
- Hint
  - Find a way to modify the plaintext of the 5th block



Topics for today

- Recap
  - Block ciphers
  - Block cipher modes
  - ECB weaknesses
- Block cipher modes
  - Exploiting the ECB's weakness (Micro-labs)?
  - Cipher block chain (CBC)
  - Exploiting the CBC's weakness (Micro-labs)?
  - How can we address the CBC's weakness?

Thank You!

Tu/Th 4:00 – 5:50 PM

Sanghyun Hong

sanghyun.hong@oregonstate.edu

Oregon State University

SAIL

Secure AI Systems Lab