ENCS5121
Information Security and
Computer Networks Laboratory

EXPERIMENT #3
Padding Oracle Attack

Slides by: Mohamad Balawi
Updated By: Tariq Odeh

BIRZEIT UNIVERSITY

Overview

- **Objective:** Gain hands-on experience with the Padding Oracle Attack.
- **Scenario:**
  - Systems verify padding in ciphertext and throw errors if invalid.
  - This seemingly harmless behavior enables an attack to uncover secret messages.
- **Lab Setup:**
  - Two oracle servers, each hiding a secret message.
  - Given: Ciphertext and IV (Initialization Vector).
- **Task:** Use oracle's padding validation responses to discover the secret message.
- **Topics Covered:**
  - Secret-key encryption
  - Encryption modes & padding
  - Padding Oracle Attack

Outline

- **Lab Environment**
- **Task 1: Getting Familiar with Padding.**
- **Task 2: Padding Oracle Attack (Level 1).**
- **Task 3: Padding Oracle Attack (Level 2).**

Lab Environment

جامعة بيرزيت BIRZEIT UNIVERSITY

Container Setup and Commands

Setup Instructions:

- Download Labsetup.zip to your VM.
  - (https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Padding_Oracle/)
- Unzip and enter the Labsetup folder.
- Use docker-compose.yml to set up the lab environment

```
$ docker-compose build  # Build the container images
$ docker-compose up     # Start the containers
$ docker-compose down   # Shut down the containers

// Aliases for the Compose commands above
$ dcbuild       # Alias for: docker-compose build
$ dcup          # Alias for: docker-compose up
$ dcdown        # Alias for: docker-compose down
```

Container Setup and Commands (Cont.)

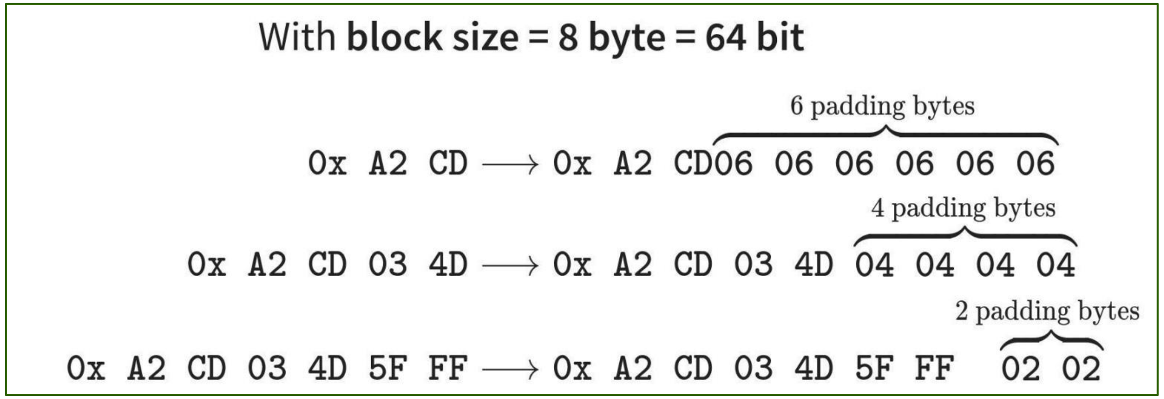
```
$ dockps         // Alias for: docker ps --format "{{.ID}}  {{.Names}}"
$ docksh <id>    // Alias for: docker exec -it <id> /bin/bash

// The following example shows how to get a shell inside hostC
$ dockps
b1004832e275  hostA-10.9.0.5
0af4ea7a3e2e  hostB-10.9.0.6
9652715c8e0a  hostC-10.9.0.7

$ docksh 96
root@9652715c8e0a:/#

// Note: If a docker command requires a container ID, you do not need to
//       type the entire ID string. Typing the first few characters will
//       be sufficient, as long as they are unique among all the containers.
```

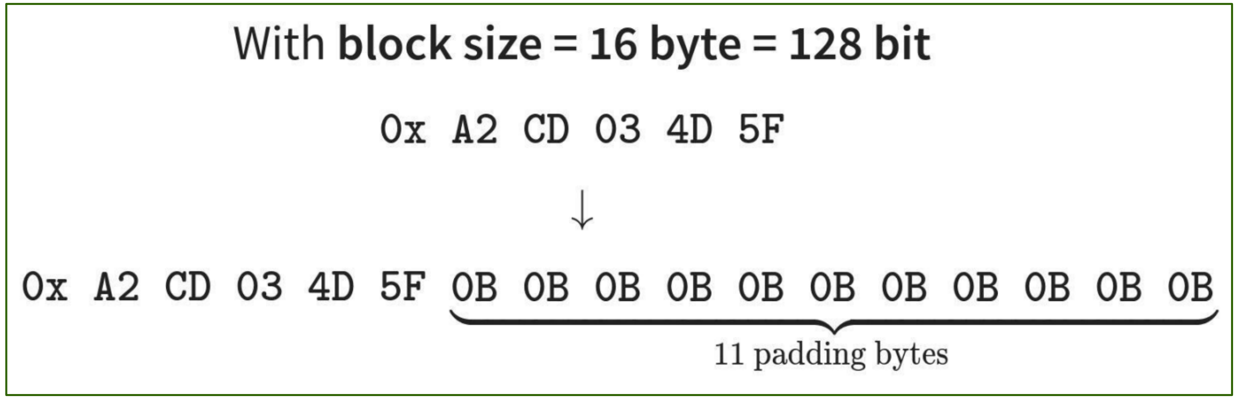
Task 1: Getting Familiar with Padding

BIRZEIT UNIVERSITY

PKCS#7

- **Described in RFC 5652, PKCS #7 works as follows:**
  - The value of each added byte is the number of bytes that are added.





Experiment with different file sizes

Steps	Explanation	Command
1	**Create files with 5, 10, and 16 bytes**	echo -n "12345" > f1.txt # 5 bytes echo -n "1234567890" > f2.txt # 10 bytes echo -n "1234567890123456" > f3.txt # 16 bytes
2	**Encrypt the files using AES-128-CBC**	openssl enc -aes-128-cbc -e -in f1.txt -out f1.enc openssl enc -aes-128-cbc -e -in f2.txt -out f2.enc openssl enc -aes-128-cbc -e -in f3.txt -out f3.enc
3	**Decrypt the files with padding disabled**	openssl enc -aes-128-cbc -d -nopad -in f1.enc -out f1_decrypted.txt openssl enc -aes-128-cbc -d -nopad -in f2.enc -out f2_decrypted.txt openssl enc -aes-128-cbc -d -nopad -in f3.enc -out f3_decrypted.txt
4	**Display the files in hexadecimal format**	xxd f1_decrypted.txt xxd f2_decrypted.txt xxd f3_decrypted.txt

Task 2: Padding Oracle Attack (Level 1)

جامعة بيرزيت BIRZEIT UNIVERSITY

The Oracle Setup

```
$ nc 10.9.0.80 5000

0102030405060708010203040506070 8a9b2554b0944118061212098f2f238cd779ea0aae3d9d020f3677bfcb3cda9ce
```

```
# 96 hex digits (48 bytes)

# IV (16-byte)                     # C1 (16-byte)                     # C2 (16-byte)
01020304050607080102030405060708 a9b2554b0944118061212098f2f238cd 779ea0aae3d9d020f3677bfcb3cda9ce
```

Useful Notation

Plaintext blocks denoted with $P_1, P_2, \ldots, P_m$.

Ciphertext blocks denoted with $C_1, C_2, \ldots, C_m$.

Where m denotes the total number of blocks.

$$P_i^j := j\text{-th byte of the } i\text{-th plaintext block}$$

$$C_i^j := j\text{-th byte of the } i\text{-th ciphertext block}$$

Let n denote the byte length of the block cipher in use.

$$P_1 := P_1^1 \, , \, P_1^2 \, , \, \ldots \, , \, P_1^n$$

$$\downarrow$$

$$C_1 := C_1^1 \, , \, C_1^2 \, , \, \ldots \, , \, C_1^n$$

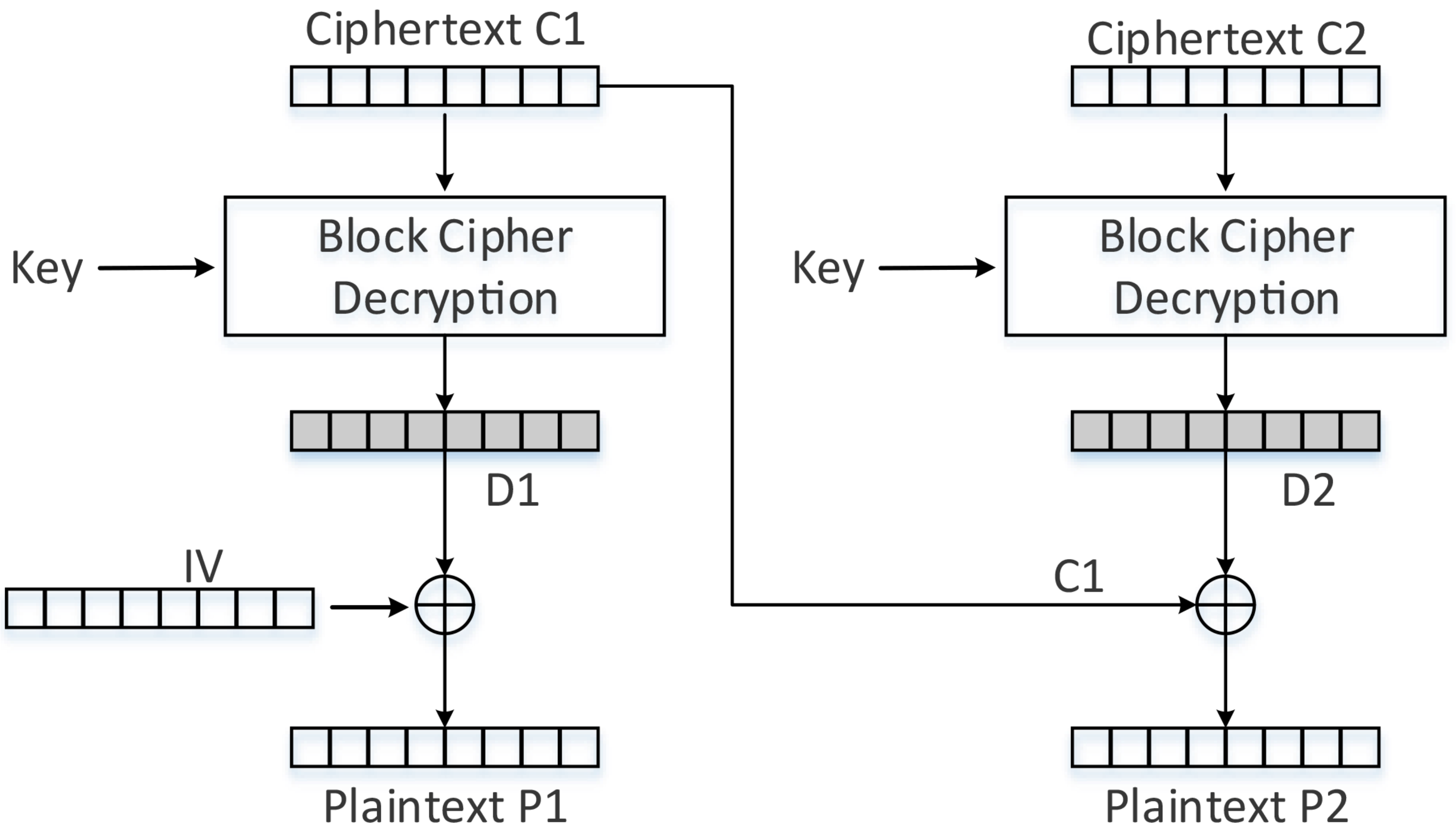
Useful Notation (Cont.)

Mathematically, let C be the ciphertext of the plaintext P. Then

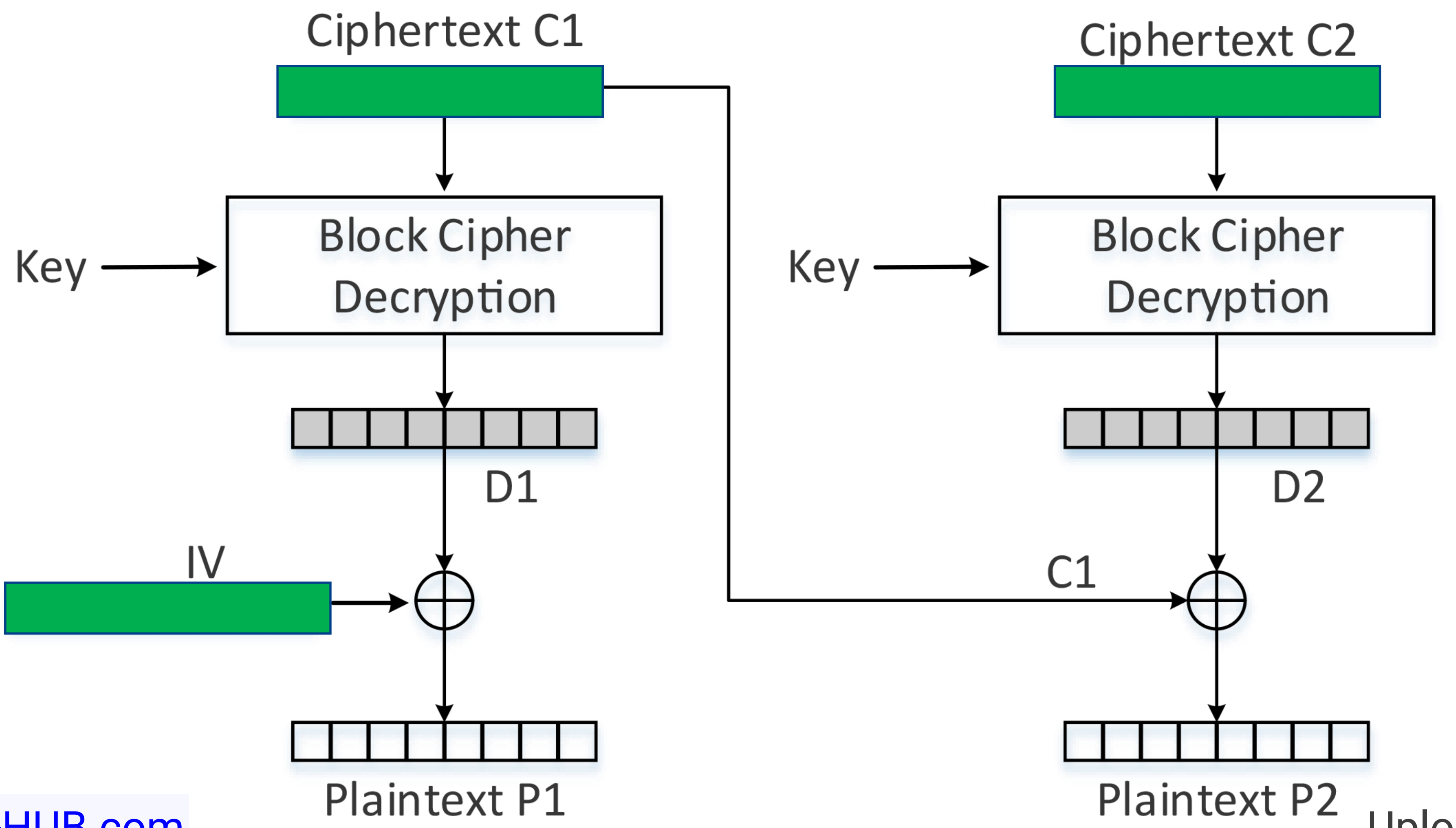
check if the ciphertext is valid or not

$$O(C) = \begin{cases} 1 & , \quad P \text{ is correctly padded according to PKCS\#7} \\ 0 & , \quad \text{otherwise} \end{cases}$$

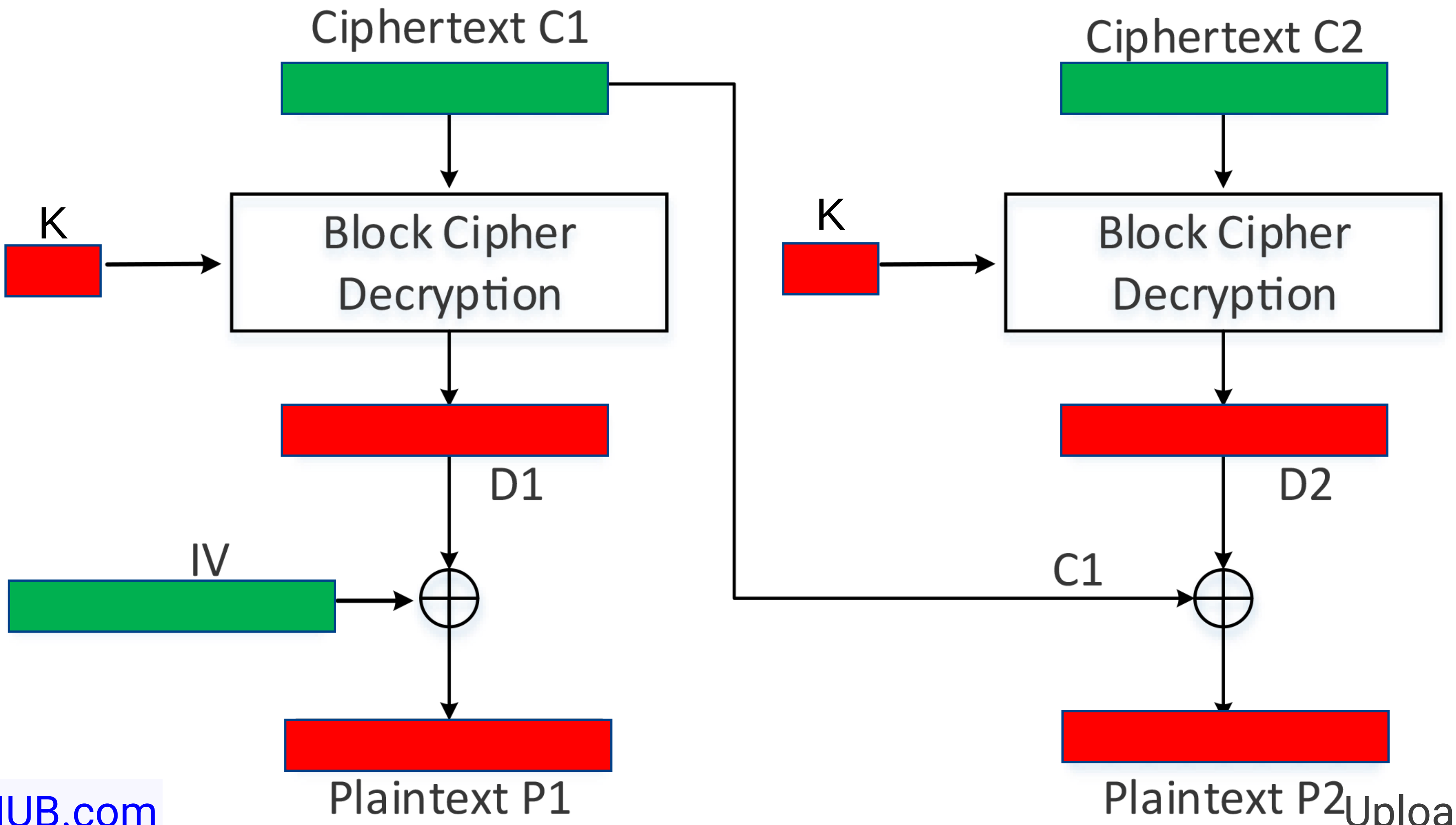
CBC Mode


Ciphertext C1
Ciphertext C2
Key
Block Cipher Decryption
Key
Block Cipher Decryption
D1
D2
IV
C1
Plaintext P1
Plaintext P2

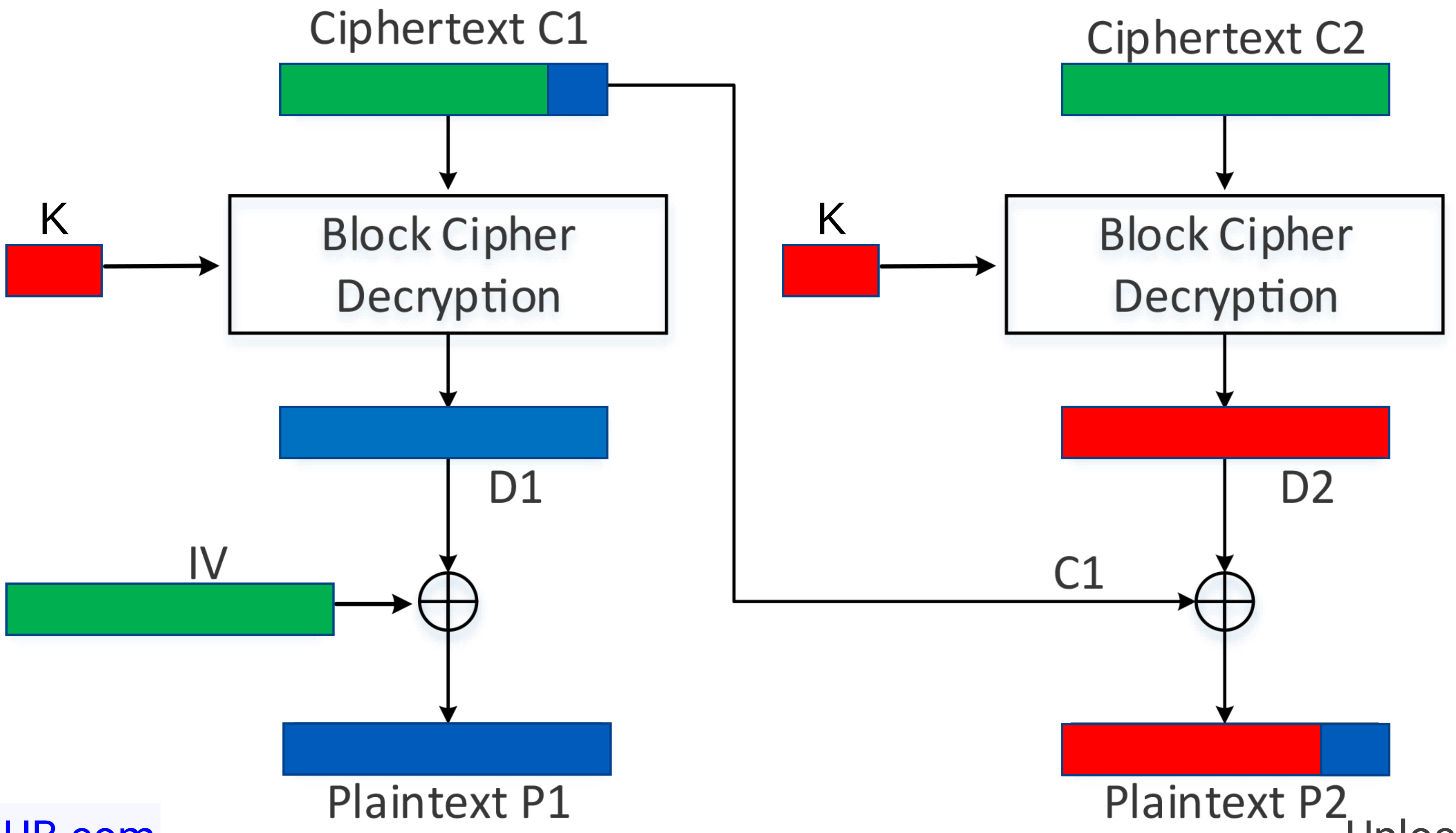
Green means we know this information



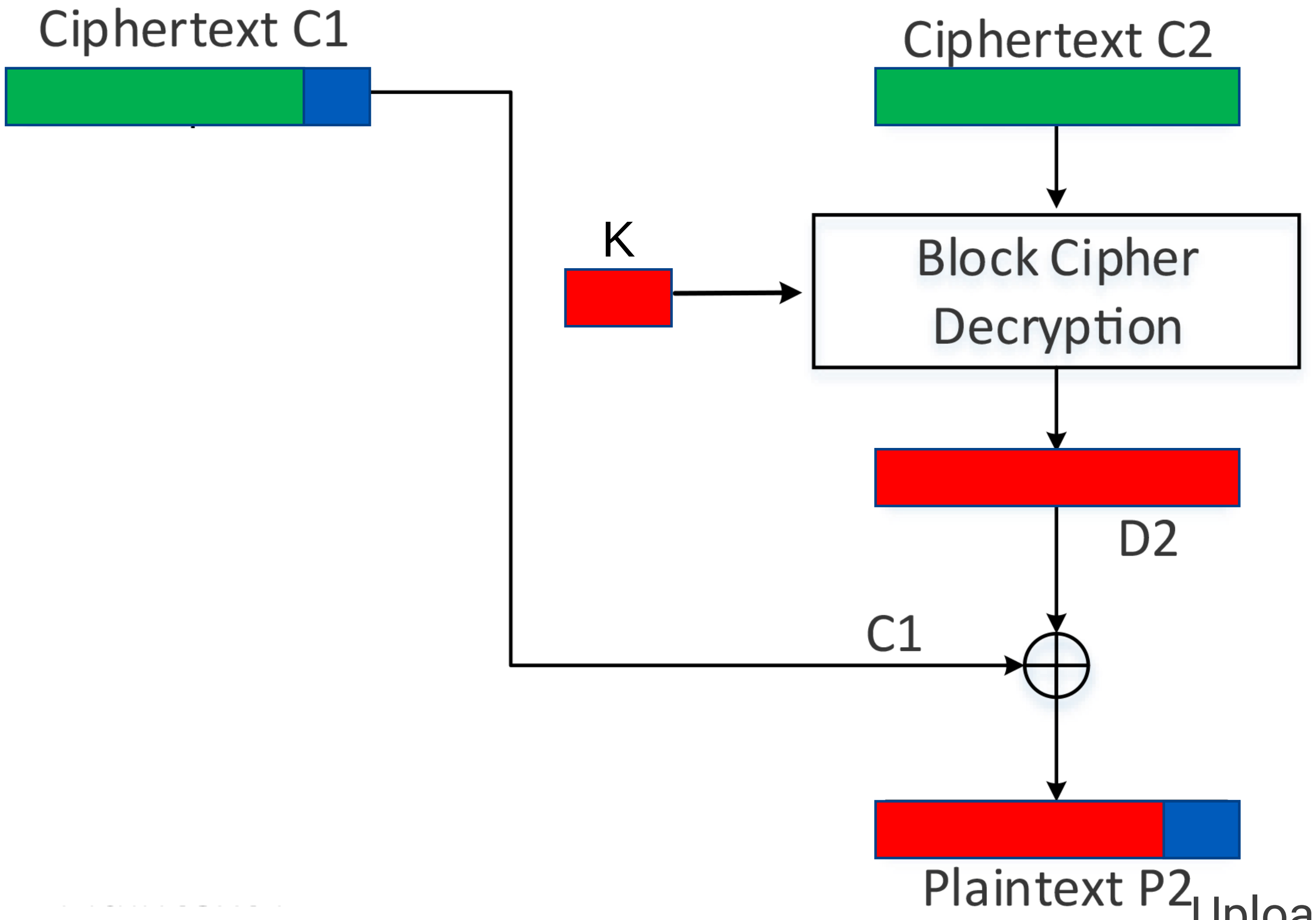
Red means we don't know this information



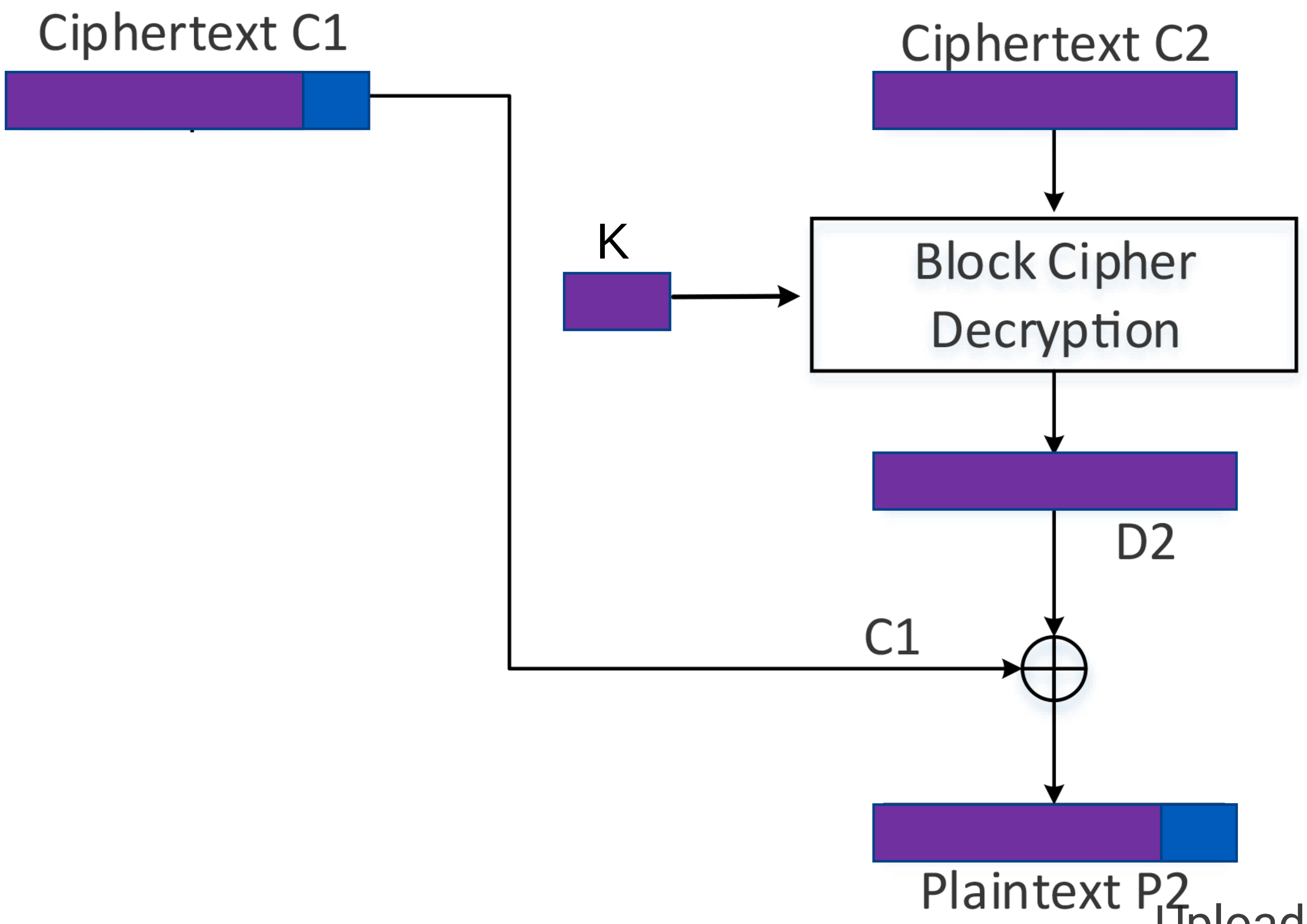
Blue means information has changed



We don't care about the first block since it does not have any padding



Purple means it is the same before and after our modification



We will find the last byte of P_2 using the **oracle** exposed by the server in a process of **trial and error.**

- Is the last byte of P_2 equal to `0x00`?
- Is the last byte of P_2 equal to `0x01`?
- ...
- Is the last byte of P_2 equal to `0xFF`?

Consider then the following question

Q: Is the last byte of P_2 equal to `0x41`?

Using our new notation we can rephrase the question as follows

Q: $P_2^n = \texttt{0x41}$?

The idea is to start from C_1 and construct a new $\hat{C}_1$

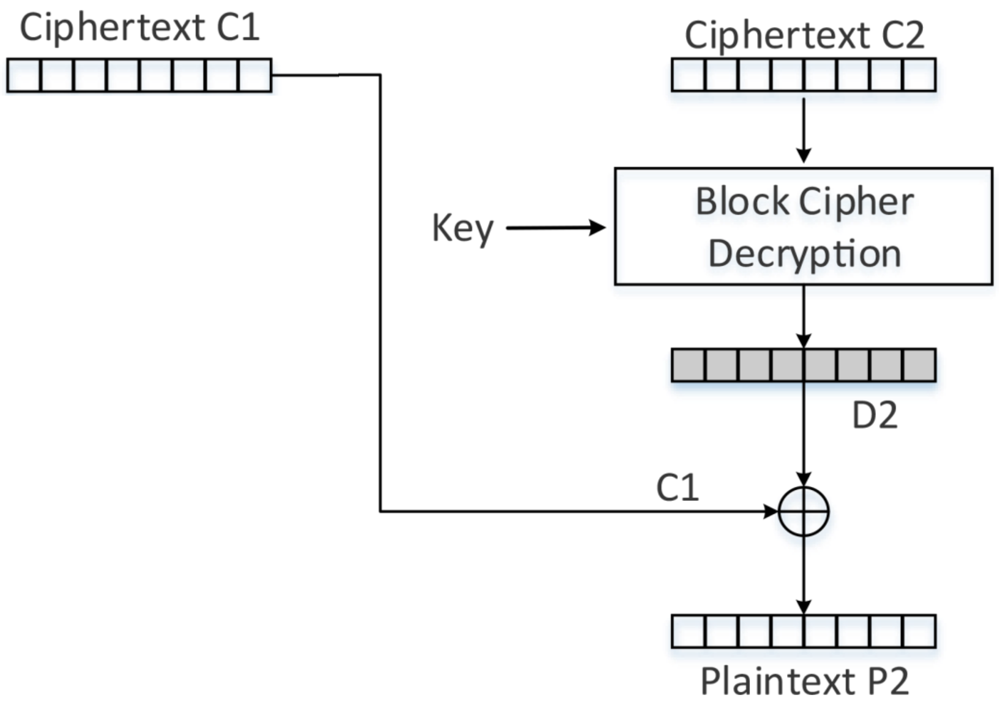
$$\hat{C}_1 := C_1^1 \, , \, C_1^2 \, , \, C_1^3 \, , \, \ldots \, , \, C_1^{n-1} \, , \, \underbrace{C_1^n \oplus \texttt{0x41} \oplus \texttt{0x01}}_{\text{byte changed}}$$

Suppose now the attacker sends this new ciphertext $\hat{C}$ to the oracle, and the oracle replies with

$$O(\hat{C}) = 1$$

That is, the associated plaintext $\hat{P}$ is correctly padded according to **PKCS#7**

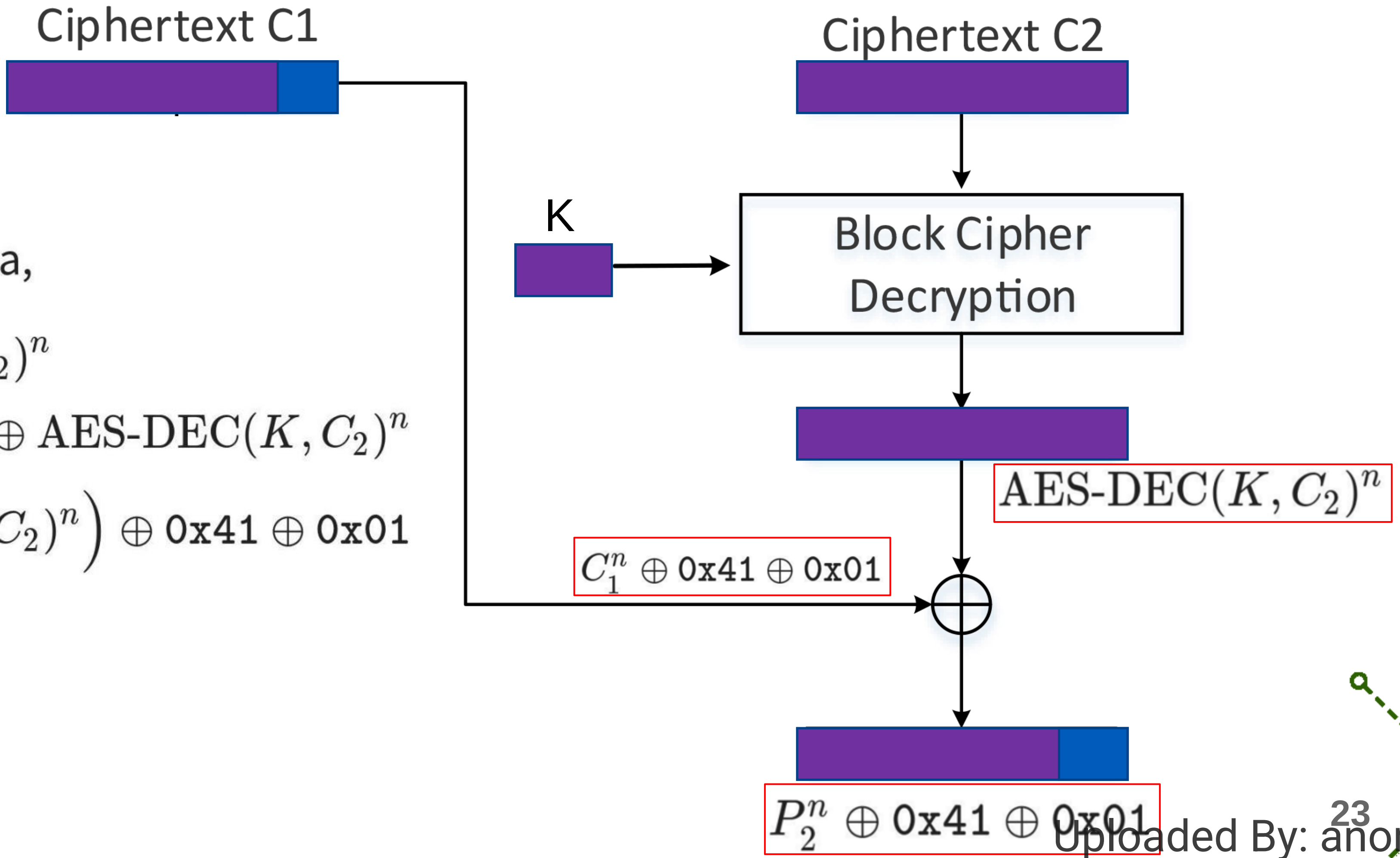
What can we infer?



Purple means it is the same before and after our modification

In formula,

$$\hat{P}_2^n = \hat{C}_1^n \oplus \text{AES-DEC}(K, C_2)^n$$
$$= \left(C_1^n \oplus \texttt{0x41} \oplus \texttt{0x01}\right) \oplus \text{AES-DEC}(K, C_2)^n$$
$$= \left(C_1^n \oplus \text{AES-DEC}(K, C_2)^n\right) \oplus \texttt{0x41} \oplus \texttt{0x01}$$
$$= P_2^n \oplus \texttt{0x41} \oplus \texttt{0x01}$$



Therefore,

$$O(\hat{C}) = 1 \implies P_2^n = \texttt{0x41} \text{ highly likely}$$

$$O(\hat{C}) = 0 \implies P_2^n \neq \texttt{0x41}$$

Q: $P_2^{n-1} = \texttt{0x41}$?

The construction of $\hat{C}_1$ is done as follows

$$\hat{C}_1 := C_1^1 \, , \, C_1^2 \, , \, C_1^3 \, , \, \ldots \, , \, \underbrace{C_1^{n-1} \oplus \texttt{0x41} \oplus \texttt{0x02}}_{\text{byte changed}} \, , \, \underbrace{C_1^n \oplus P_2^n \oplus \texttt{0x02}}_{\text{byte changed}}$$

Q: $P_2^{n-2} = \texttt{0x41}$?

$$\hat{C}_1 := C_1^1 \, , \, C_1^2 \, , \, C_1^3 \, , \, \ldots \, , \, \underbrace{C_1^{n-2} \oplus \texttt{0x41} \oplus \texttt{0x03}}_{\text{byte changed}} \, , \, \underbrace{C_1^{n-1} \oplus P_2^{n-1} \oplus \texttt{0x03}}_{\text{byte changed}} \, , \, \underbrace{C_1^n \oplus P_2^n \oplus \texttt{0x03}}_{\text{byte changed}}$$

Padding Oracle Attack Visualization

https://www.youtube.com/watch?v=uDHo-UAM6_4&ab_channel=PranavJain

https://paddingoracle.github.io/

Task 3: Padding Oracle Attack (Level 2)

BIRZEIT UNIVERSITY

First Iteration

Q: $P_2^n = \texttt{0x41}$?

The idea is to start from C_1 and construct a new $\hat{C}_1$

$$\hat{C}_1 := C_1^1, C_1^2, C_1^3, \ldots, C_1^{n-1}, \underbrace{C_1^n \oplus \texttt{0x41} \oplus \texttt{0x01}}_{\text{byte changed}}$$

Found using python code $\hat{C}_1^n = 0xCF$

Assumption $\hat{C}_1^n = C_1^n \oplus P_2^n \oplus 0x01$

$0xCF = C_1^n \oplus P_2^n \oplus 0x01$

$0xCE = C_1^n \oplus P_2^n$

Given from the server $C_1^n = 0xCD$

$0xCE = 0xCD \oplus P_2^n$

$P_2^n = 0x03$

Second Iteration

Q: $P_2^{n-1} = \texttt{0x41}$?

The construction of $\hat{C}_1$ is done as follows

$$\hat{C}_1 := C_1^1 , C_1^2 , C_1^3 , \ldots , \underbrace{C_1^{n-1} \oplus \texttt{0x41} \oplus \texttt{0x02}}_{\text{byte changed}} , \underbrace{C_1^n \oplus P_2^n \oplus \texttt{0x02}}_{\text{byte changed}}$$

$$\hat{C}_1^n = C_1^n \oplus P_2^n \oplus 0\text{x}02$$

From first iteration $C_1^n = 0xCD$

From first iteration $P_2^n = 0x03$

$$\hat{C}_1^n = 0xCD \oplus 0x03 \oplus 0\text{x}02$$

$$\hat{C}_1^n = 0xCC$$

Found using python code $\hat{C}_1^{n-1} = 0x39$

$$\hat{C}_1^{n-1} = C_1^{n-1} \oplus P_2^{n-1} \oplus 0\text{x}02$$

$$0x39 = C_1^{n-1} \oplus P_2^{n-1} \oplus 0\text{x}02$$

From the server $C_1^{n-1} = 0x38$

$$0x39 = 0\text{x}38 \oplus P_2^{n-1} \oplus 0\text{x}02$$

$$P_2^{n-1} = 0x03$$

Third Iteration

$$\textbf{Q: } P_2^{n-2} = \texttt{0x41}?$$

$$\hat{C}_1 := C_1^1\ ,\ C_1^2\ ,\ C_1^3\ ,\ \ldots\ ,\ \underbrace{C_1^{n-2} \oplus \texttt{0x41} \oplus \texttt{0x03}}_{\text{byte changed}}\ ,\ \underbrace{C_1^{n-1} \oplus P_2^{n-1} \oplus \texttt{0x03}}_{\text{byte changed}}\ ,\ \underbrace{C_1^{n} \oplus P_2^{n} \oplus \texttt{0x03}}_{\text{byte changed}}$$

$$\hat{C}_1^{n} = C_1^{n} \oplus P_2^{n} \oplus 0\mathrm{x}03$$
$$\hat{C}_1^{n} = 0xCD \oplus 0x03 \oplus 0\mathrm{x}03$$
$$\hat{C}_1^{n} = 0xCD$$

$$\hat{C}_1^{n-1} = C_1^{n-1} \oplus P_2^{n-1} \oplus 0\mathrm{x}03$$
$$\hat{C}_1^{n-1} = 0x38 \oplus 0x03 \oplus 0\mathrm{x}03$$
$$\hat{C}_1^{n-1} = 0x38$$

Found using python code $\hat{C}_1^{n-2} = 0xF2$

$$\hat{C}_1^{n-2} = C_1^{n-2} \oplus P_2^{n-2} \oplus 0\mathrm{x}03$$

From the server $C_1^{n-2} = 0xF2$

$$0xF2 = 0\mathrm{xF}2 \oplus P_2^{n-2} \oplus 0\mathrm{x}03$$
$$P_2^{n-2} = 0x03$$

References

- **SEED LABS .**
- **Some slides are from the following sources:**
  - **Leonardo Tamiano**
  - **(CNS Lab 03 – Padding Oracle On AES-CBC-PKCS#7)**