

Faculty of Engineering & Technology

Electrical & Computer Engineering Department

APPLIED CRYPTOGRAPHY- ENCS4320

Report for Project

**Crypto Lab—Padding Oracle Attack**

Prepared by:

Ahmad Zubaidia 1200105

Partners: there is no partners.

Instructor: Dr. Ahmad Alsadeh

Section: 1

Date:26/6/2023

# ABSTRACT

The purpose of this experiment, was to study the details of cryptographic padding, especially PKCS#5, and its importance in protecting data using block ciphers. We developed a simulation intended to simulate actual encryption situations using Docker, a platform that makes it easier to create regulated and consistent environments. We started out by looking at the core concepts of PKCS#5 padding and its function in block cipher encryption. After that, we focused our attention to a serious flaw caused by incorrect implementation known as the Padding Oracle Attack. We manually carried out this attack using careful, step-by-step analysis, modifying padding data to decrypt encrypted information. The experiment's next step included speeding the procedure by creating an automated attack program. As a result, we were able to observe and evaluate the effectiveness and speed of a Padding Oracle Attack when it executed out automatically. In the experiment's final stages, the Padding Oracle Attack was used to enable the decoding of a fictitious secret message without the need for prior knowledge of the encryption key. This showed the potential consequences of cryptographic padding issues. This experiment highlighted the value of strong encryption techniques for preventing weaknesses while offering important insights into the area of cryptography.

Table of Contents

[ABSTRACT I](#_Toc138648693)

[Table of figures III](#_Toc138648694)

[Introduction: 1](#_Toc138648695)

[Procedure 2](#_Toc138648696)

[Task 1 2](#_Toc138648697)

[What Was Done: 4](#_Toc138648698)

[Observations: 4](#_Toc138648699)

[Explanation of Observations: 4](#_Toc138648700)

[Task 2 5](#_Toc138648701)

[What Was Done: 8](#_Toc138648702)

[Observations: 8](#_Toc138648703)

[Explanation of Observations: 8](#_Toc138648704)

[Task 3 9](#_Toc138648705)

[Using port 6000 11](#_Toc138648706)

[The output: 14](#_Toc138648707)

[What Was Done: 19](#_Toc138648708)

[Observations: 19](#_Toc138648709)

[Explanation of Observations: 19](#_Toc138648710)

[Conclusion 20](#_Toc138648711)

# Table of figures

[Figure 1: makeing the first file and see padding 2](#_Toc138648728)

[Figure 2: setting up the servers 2](#_Toc138648729)

[Figure 3 : task 1 , 5 bytes 3](#_Toc138648730)

[Figure 4:task 1 , 10 bytes 3](#_Toc138648731)

[Figure 5: task1 , 16 bit 3](#_Toc138648732)

[Figure 6: iteration ,K=1, D[15]. , K=2 ,D[14]. 5](#_Toc138648733)

[Figure 7: the values which makes the padding valid K=1,K=2. 5](#_Toc138648734)

[Figure 8: Bytes ,K=3, D[13]. , K=4 ,D[12]. 6](#_Toc138648735)

[Figure 9:the values which makes the padding valid K=3,K=4 6](#_Toc138648736)

[Figure 10: Bytes ,K=5, D[11]. , K=6 ,D[10]. 7](#_Toc138648737)

[Figure 11: the values which makes the padding valid K=5,K=6 & P(plain text for 6 Bytes) 7](#_Toc138648738)

[Figure 12: Loop over K values from 1 to 16 9](#_Toc138648739)

[Figure 13: the values of valid padding. 10](#_Toc138648740)

[Figure 14:The Level-2 server listens to port 6000 11](#_Toc138648741)

[Figure 15: Cypher from port 6000 12](#_Toc138648742)

[Figure 16: Cypher Blocks with IV 12](#_Toc138648743)

[Figure 17: Automated attack process for three blocks. 13](#_Toc138648744)

[Figure 18: three blocks’ cyphers 14](#_Toc138648745)

[Figure 19: for Block one Valid padding values 15](#_Toc138648746)

[Figure 20: For Block Two 16](#_Toc138648747)

[Figure 21: For Block Three 17](#_Toc138648748)

[Figure 22: the plain text of three blocks in hex 18](#_Toc138648749)

[Figure 23: text of plain text 18](#_Toc138648750)

# Introduction:

In today’s world, almost everything we do involves computers and the internet. From sending emails and sharing pictures to storing important documents, we rely heavily on digital technology. However, as we all know, the online world isn't always safe, and that's why we need ways to protect our information from prying eyes. This is where cryptography comes in.

Cryptography is like a digital lock and key system that helps keep our data safe. It’s a way to change the information into a secret code that only someone with the right key can understand. One of the ways cryptography does this is through something called block ciphers. Imagine you have a long message you want to secure. Block ciphers take this message and break it into smaller, equal-sized pieces, called blocks. These blocks are then scrambled up with a key so that no one can read them unless they have the key to unscramble them.

But what happens if the last piece of your message isn't big enough to make a full block? This is where PKCS#5 padding steps in. PKCS#5 padding adds extra bits to the last block so that it’s the right size. Think of it as adding extra stuffing to a package so it fits snugly in its box.

Now, you might think that with all this scrambling and padding, your message is safe. However, like any lock, there are people who try to pick it. In cryptography, one way they do this is through something called a Padding Oracle Attack. This is a sneaky way for a hacker to figure out the padding pattern used in the last block of the message, and eventually, decode the entire message without needing the key!

Understanding the details of how PKCS#5 padding works and how it can be exploited through the Padding Oracle Attack is important. It helps us build more secure systems and teaches us how to protect sensitive data.

In this experiment, we will take a close look at PKCS#5 padding, learn how it adds those extra bits to make the data just the right size, and how this method can sometimes be used against us through the Padding Oracle Attack. We will also try to understand how these attacks work and think about ways we can defend against them.

By diving deep into these topics, we're not just learning about codes and ciphers. We are contributing to a safer and more secure digital world for everyone.

# Procedure

## Task 1

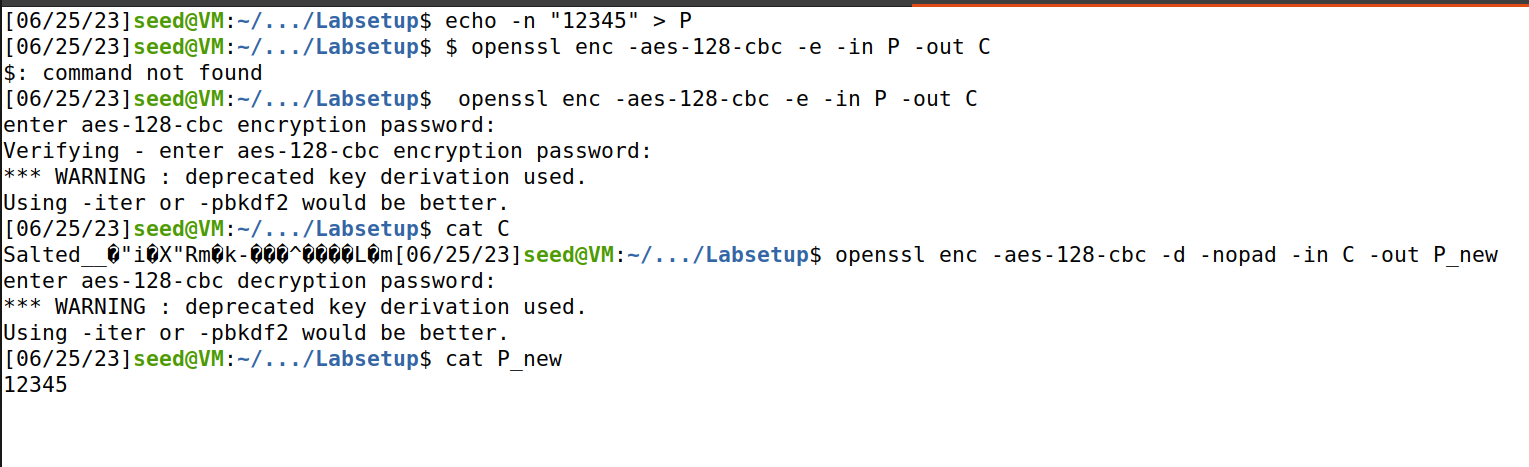


Figure 1: makeing the first file and see padding

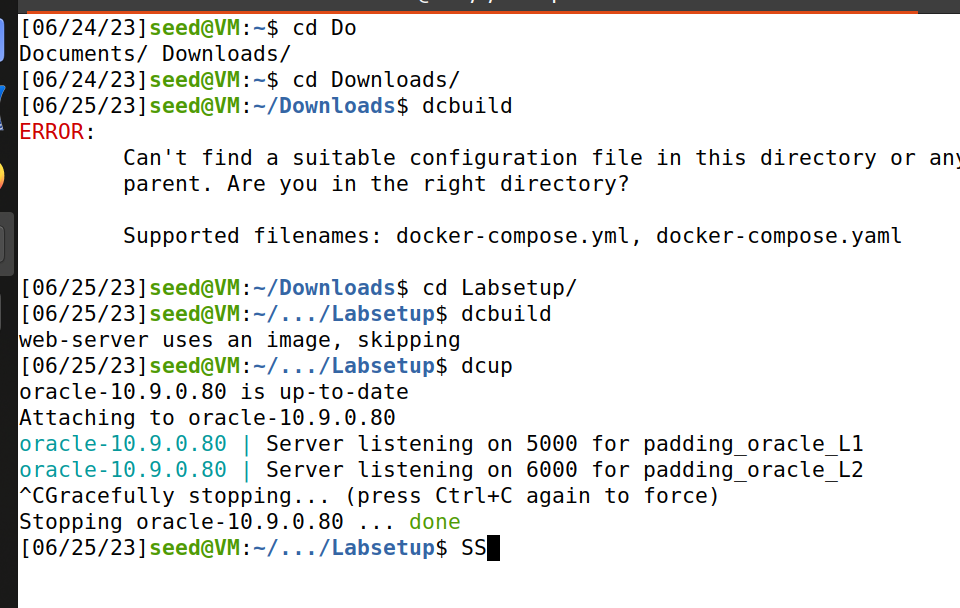


Figure 2: setting up the servers

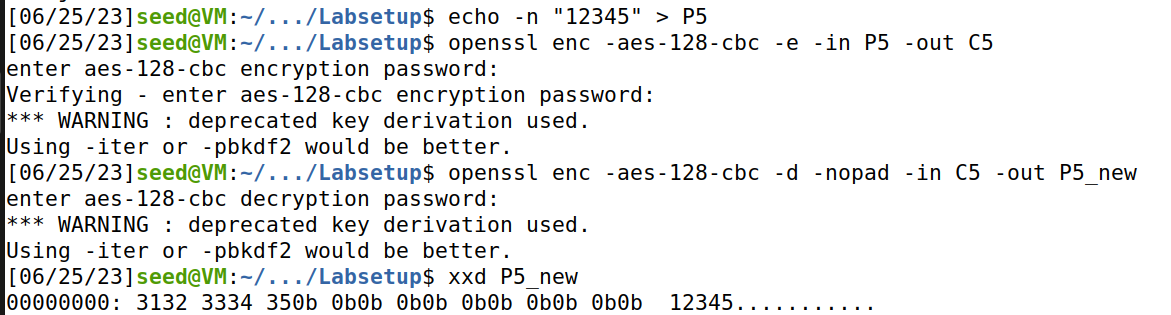


Figure 3 : task 1 , 5 bytes

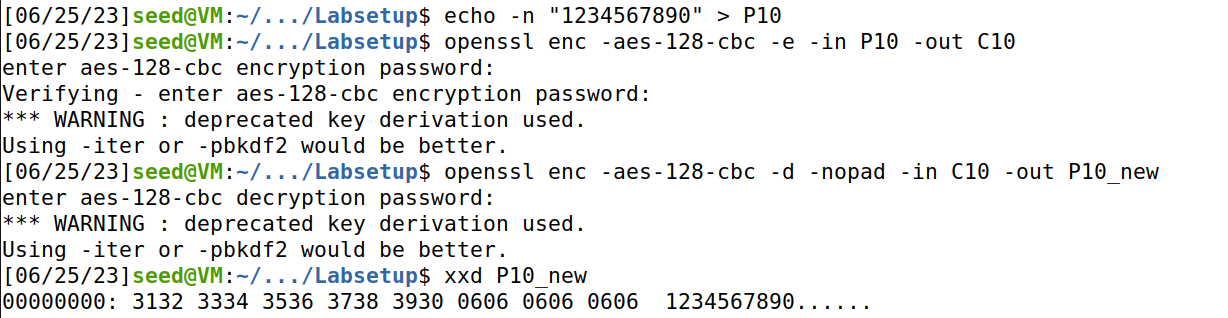


Figure 4:task 1 , 10 bytes

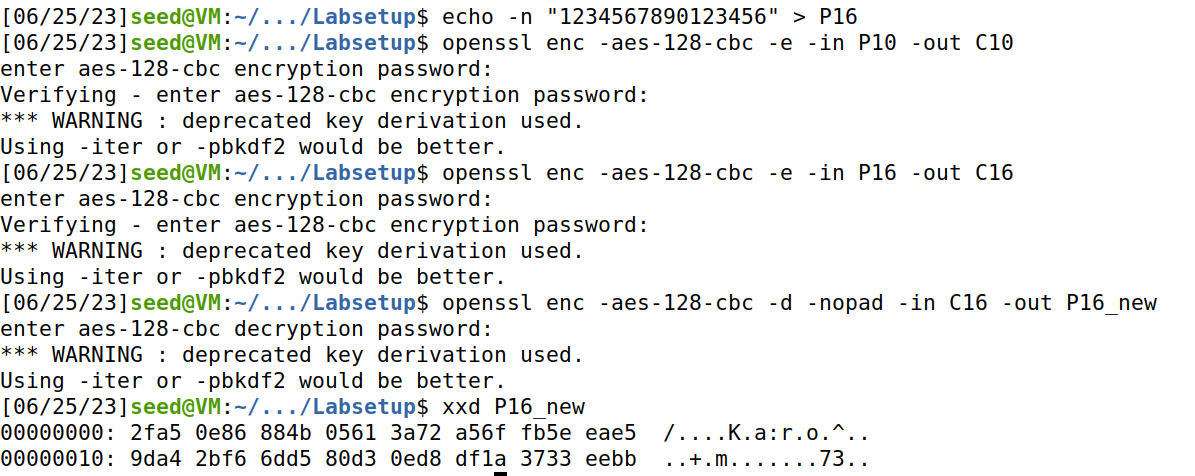


Figure 5: task1 , 16 bit

### What Was Done:

The objective of Task 1 was to become familiar with how padding, specifically PKCS#5 padding, functions in block ciphers. For block ciphers, padding may be required when the size of the plaintext is not a multiple of the block size. In this task, three files were created with sizes of 5 bytes, 10 bytes, and 16 bytes, respectively. The OpenSSL command-line tool was utilized for encrypting and decrypting these files with 128-bit AES in CBC mode. This was done to observe what data was added during the padding process.

The following commands were executed:

Created a file with 5 bytes: **echo -n "12345" > P**

Encrypted the file: **openssl enc -aes-128-cbc -e -in P -out C**

Decrypted the file without removing the padding: **openssl enc -aes-128-cbc -d -nopad -in C -out P\_new**

Displayed the content in hex format: **xxd P\_new**

These steps were repeated for files containing 10 bytes and 16 bytes.

### Observations:

Upon decrypting and examining the contents of the file **P\_new** in hex format, it was noted that additional bytes were appended to the original data. The padded data was in a non-printable form. For the file with 5 bytes, the padding was 11 bytes long, for the file with 10 bytes, the padding was 6 bytes long, and for the file with 16 bytes, no padding was added.

### Explanation of Observations:

The PKCS#5 padding scheme works by filling up the block with bytes, all of which represent the number of bytes added. For instance, in the case of the file with 5 bytes, 11 bytes of padding were added, each having the value 0x0B (which is 11 in hexadecimal). This makes the total block size 16 bytes, which is a multiple of the AES block size.

For the file with 10 bytes, 6 bytes of padding were added, each with the value 0x06.

Interestingly, for the file with exactly 16 bytes, no padding was added. This is because the data already fits perfectly into the block size required by AES.

Understanding how padding works is essential as it has implications for security and the proper functioning of cryptographic systems.

## Task 2

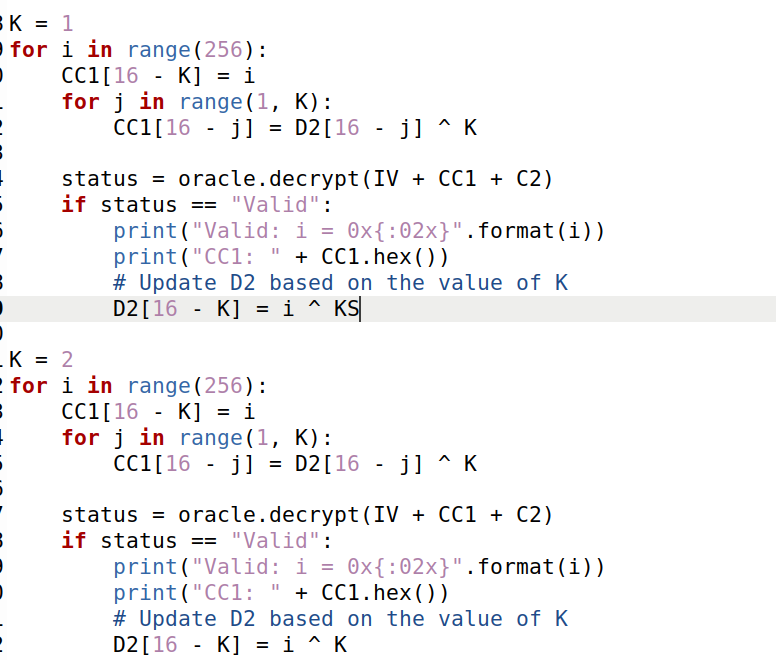


Figure 6: iteration ,K=1, D[15]. , K=2 ,D[14].

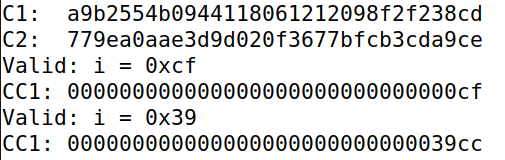


Figure 7: the values which makes the padding valid K=1,K=2.

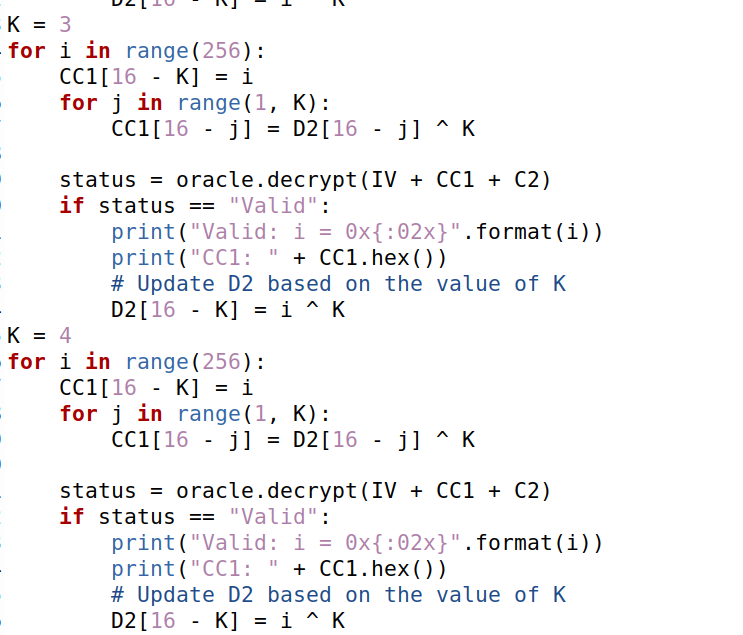


Figure 8: Bytes ,K=3, D[13]. , K=4 ,D[12].

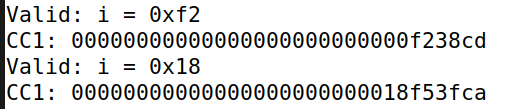


Figure 9:the values which makes the padding valid K=3,K=4

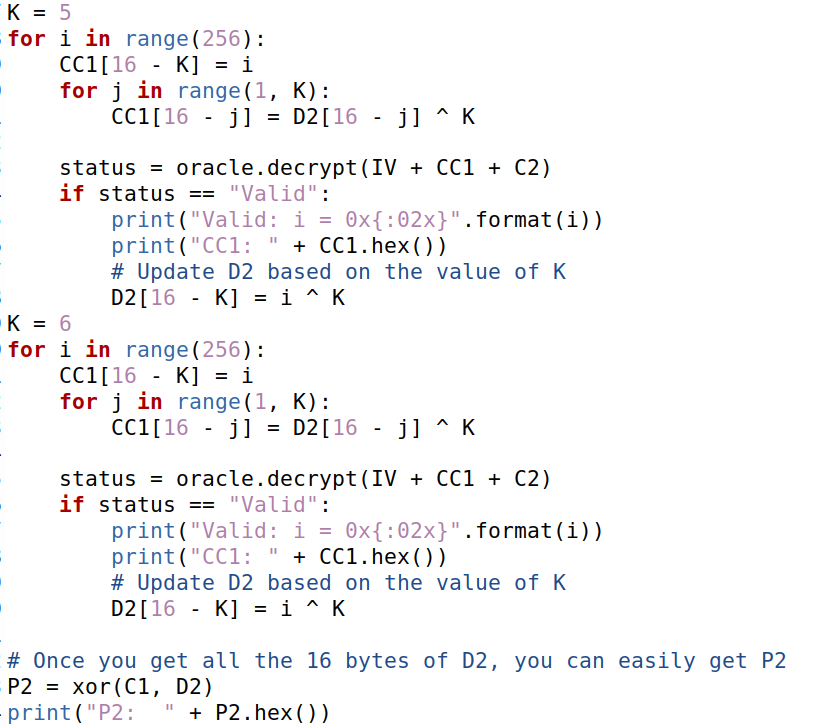


Figure 10: Bytes ,K=5, D[11]. , K=6 ,D[10].

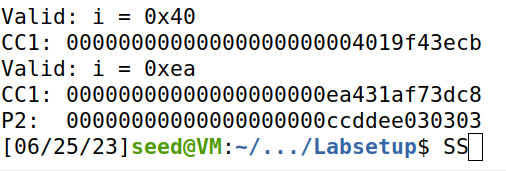


Figure 11: the values which makes the padding valid K=5,K=6 & P(plain text for 6 Bytes)

### What Was Done:

Task 2 focused on performing a padding oracle attack. Some systems, during decryption, verify the validity of the padding and return an error if the padding is invalid. This behavior can be exploited through a padding oracle attack. The goal of Task 2 was to decipher a secret message by exploiting the padding oracle without knowing the encryption key. AES-CBC was used as the encryption algorithm.

The padding oracle was set up on port 5000 and provided a ciphertext of a secret message encrypted with an unknown key K. The ciphertext consisted of an IV (initialization vector) and the encrypted message, split into blocks.

The ciphertext was fetched from the padding oracle.

Two 16-byte arrays, C1 and C2, were used to store the content of the two blocks of the ciphertext.

The padding oracle was then interacted with by sending modified versions of C1 (referred to as CC1) along with C2 and observing whether the padding was valid or not.

Through an iterative process, each byte of an array D2 was deciphered. D2 represents the output of the block cipher before XORing with the previous ciphertext block (or IV for the first block).

Upon figuring out D2, the plaintext P2 was calculated as **P2 = C1 ⊕ D2**.

### Observations:

Through multiple iterations and trying all possible values for each byte, one byte at a time, it was possible to derive values for D2 such that the padding was valid. By continuously modifying CC1 and observing the oracle's responses, the entire block D2 was eventually derived.

### Explanation of Observations:

The padding oracle attack exploits the fact that an attacker can learn whether the padding of a modified ciphertext is valid or not. By carefully choosing the modifications and observing the responses, the attacker can make educated guesses about the plaintext.

In this task, the focus was on decrypting the second block of the ciphertext. To decrypt the second block, values for D2 were needed, which could then be XORed with C1 to obtain the plaintext. By iteratively changing the bytes in CC1 and sending it with C2 to the oracle, information about the actual D2 was gathered. This iterative process was done for each byte of D2, and as the values were derived, they were used in subsequent rounds to derive the next byte.

The process was manual, and at least six bytes of D2 needed to be deciphered for the task. Once the required number of bytes for D2 was obtained, P2 could be calculated.

This task demonstrated how a seemingly harmless behavior of checking for valid padding can be exploited to derive sensitive information. It highlights the importance of careful implementation and consideration of cryptographic systems.

## Task 3

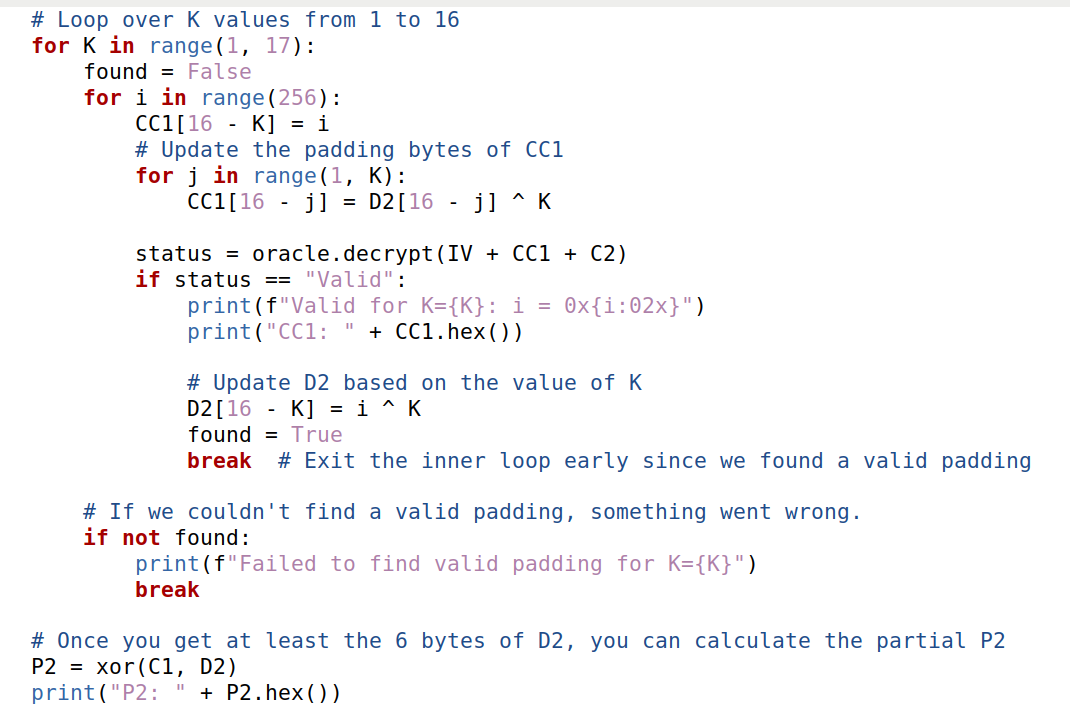


Figure 12: Loop over K values from 1 to 16

Just the reason of this section , to make the previous process, automatic , by using for loops.

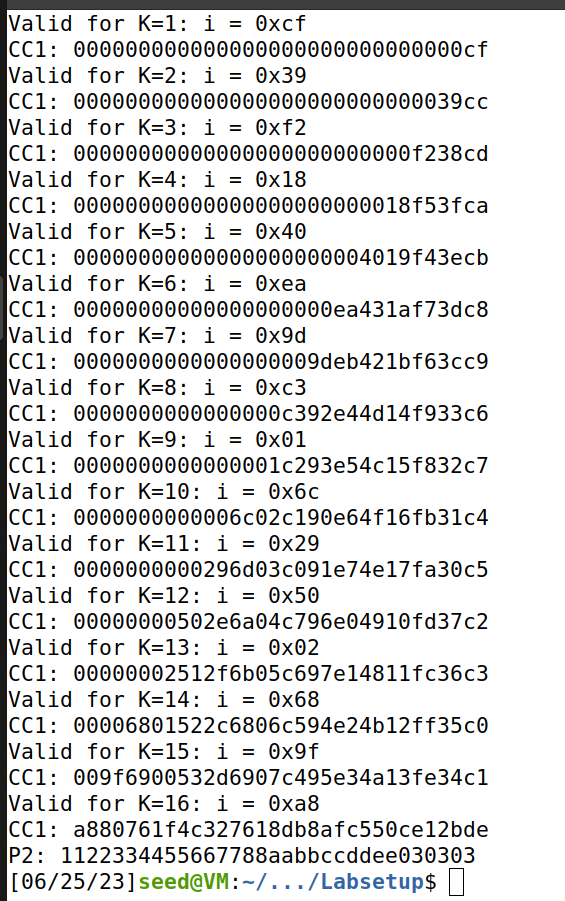
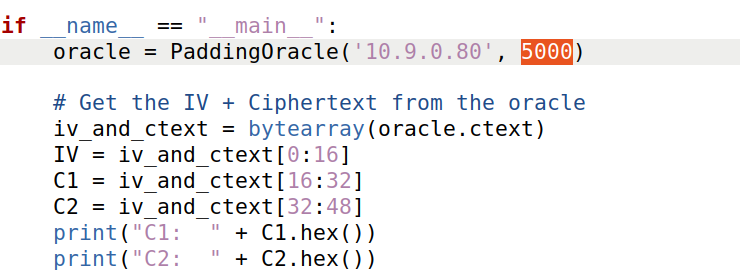


Figure 13: the values of valid padding.

### Using port 6000



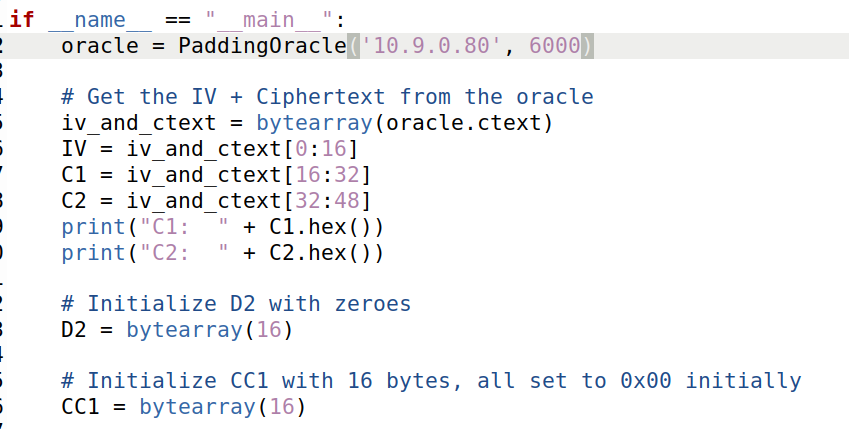


Figure 14:The Level-2 server listens to port 6000

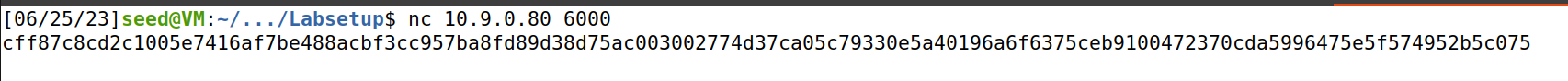


Figure 15: Cypher from port 6000

After seeing the cypher form port 6000 , we conclude some points

For this port , we got cypher which equal to 98 character which equal to 48 Byte means 3 Blocks +IV .

16 IV + 16 block 1 , 16 block 2 , 16 block 3.

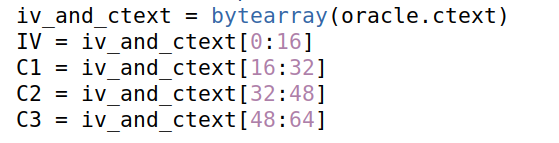
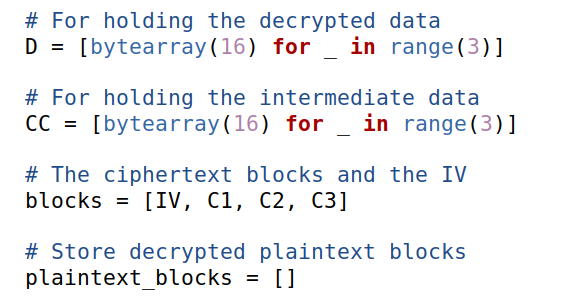
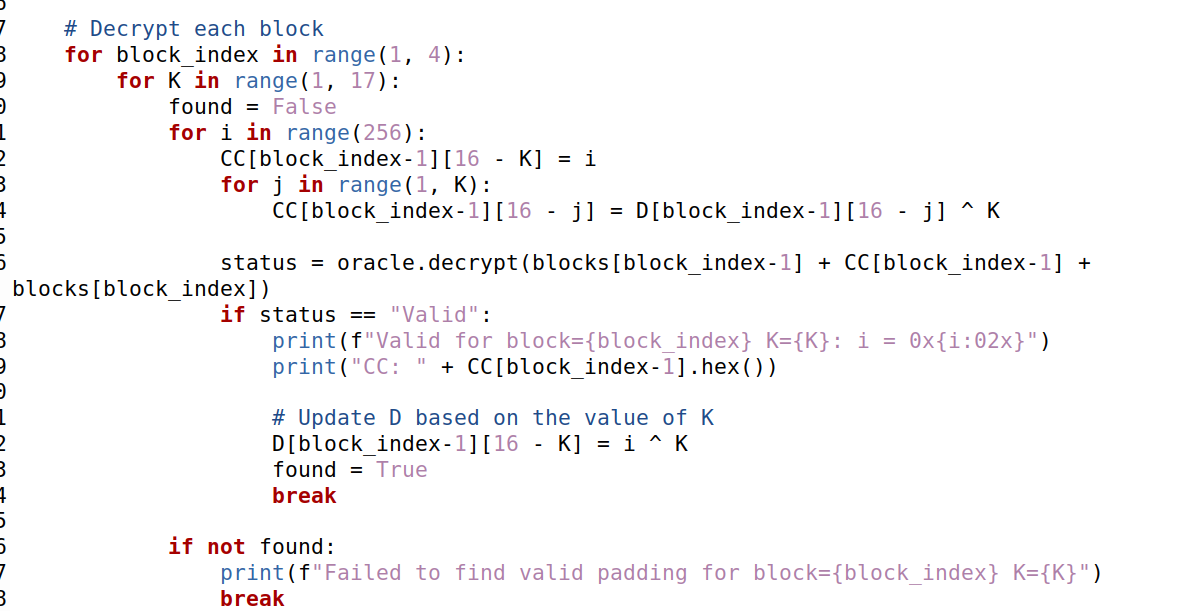


Figure 16: Cypher Blocks with IV

D1, D2 ,D3 for holding the decrypted data

And CC1, CC2, CC3 for holding the intermediate data





Firs loop for blocks , second for K (Bytes from 1-16) , the third one for itreation to find the vaild padding , the last one for updating .

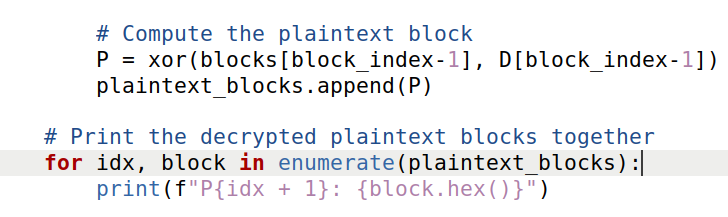


Figure 17: **Automated attack process for three blocks.**

## The output:

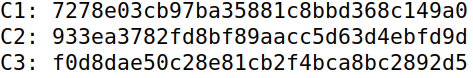


Figure 18: three blocks’ cyphers

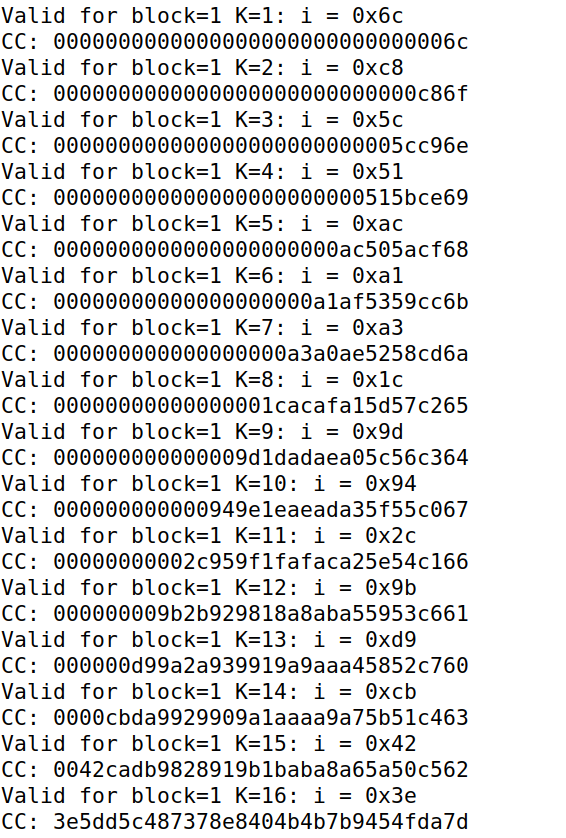


Figure 19: for Block one Valid padding values

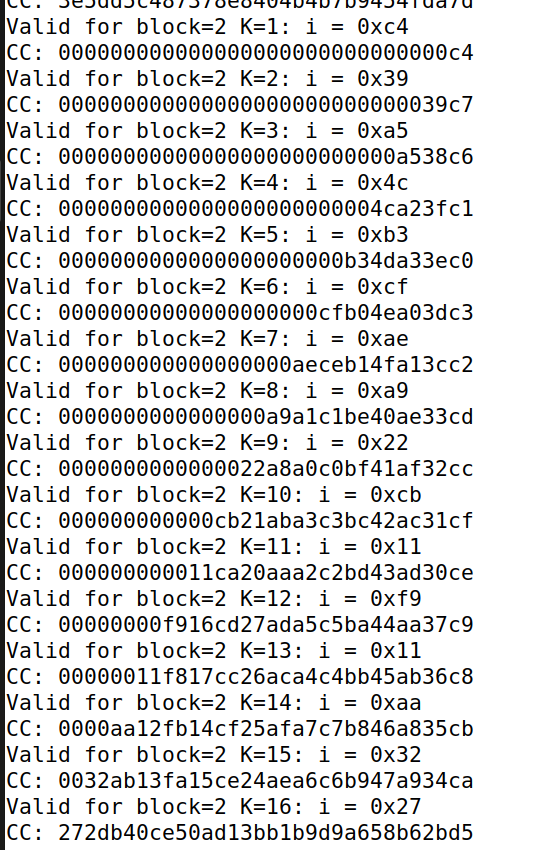


Figure 20: For Block Two

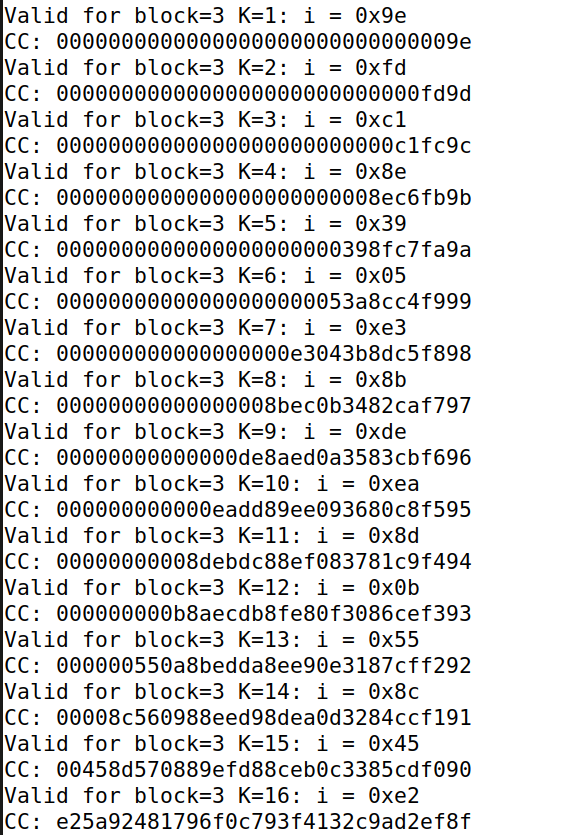


Figure 21: For Block Three

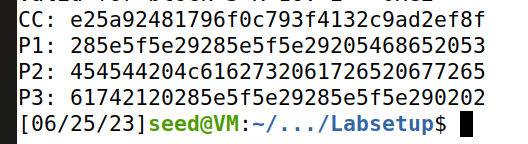


Figure 22: the plain text of three blocks in hex

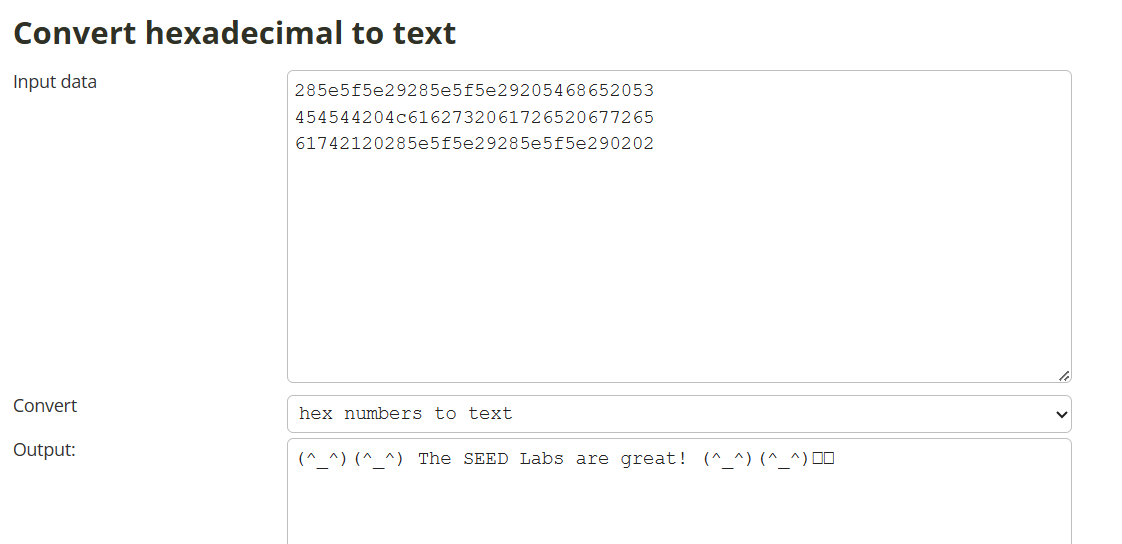


Figure 23: text of plain text

### What Was Done:

Task 3 involved automating the padding oracle attack process and extending it to decrypt all blocks of the secret message. The objective was to derive the entire plaintext by exploiting the padding oracle without prior knowledge of the encryption key. AES-CBC was again used as the encryption algorithm. A padding oracle server for Level 2 was set up on port 6000, allowing interaction for the automated attack. The server responded to queries with a ciphertext containing an IV and encrypted message blocks. The process began by fetching the ciphertext from the padding oracle. Four 16-byte arrays, including an IV and three ciphertext blocks (C1, C2, and C3), were created to store the respective components of the ciphertext. To perform the automated attack, arrays were initialized to hold the decrypted data (D) and intermediate data (CC). The blocks of the ciphertext and the IV were assigned to corresponding variables. The attack involved decrypting each block iteratively. For each block, an iterative process was conducted, focusing on one byte at a time. The goal was to modify CC1 in each round, generate a modified ciphertext, and send it to the padding oracle to determine if the padding was valid. By trying all possible byte values for each round, one byte of D2 could be derived. The code provided in the task served as a foundation for constructing the attack. The program looped through different byte values of CC1, checking the padding validity through interaction with the padding oracle. Upon finding valid padding, the derived value was used to update D2. This process was repeated until all bytes of D2 were derived for each block. Finally, the plaintext blocks were calculated by performing XOR operations between the corresponding ciphertext blocks and derived D values.

### Observations:

The automated padding oracle attack successfully decrypted all blocks of the secret message. By automating the process, it became feasible to derive the entire plaintext without manual intervention.

### Explanation of Observations:

Automating the padding oracle attack in Task 3 proved highly effective in deriving all blocks of the secret message without manual iterations. By streamlining the attack process through the use of appropriate arrays and iterative procedures for each block, the attacker systematically derived the correct values for D. The automated attack program modified CC1, checked padding validity, and updated D values accordingly. This automation showcased the scalability and efficiency of the padding oracle attack, highlighting the vulnerability of systems relying on padding validation during decryption. The successful decryption of all blocks underscores the importance of implementing robust encryption algorithms and secure padding schemes to prevent padding oracle attacks and maintain the confidentiality of sensitive information.

# Conclusion

The experiment provided important light on the function of PKCS#5 padding and the significance of block cipher encryption. The Padding Oracle Attack made it clear how cryptographic padding flaws may be used to decode data without having access to the encryption keys. The attack's human execution exposed its specifics and difficulties, but the programmed method demonstrated how automation can make such attacks more effective. This experiment emphasizes how important it is to use strong encryption procedures in order to protect data from attacks like this.