**Task1**

1.下载原始BMP图片：

wget https://hugh.myddns.me/DADA/lab2/pic\_original.bmp

2.使用ECB模式加密：

openssl enc -aes-128-ecb -e -in pic\_original.bmp -out pic\_ecb.bin -K 00112233445566778889aabbccddeeff

3.使用CBC模式加密：

openssl enc -aes-128-cbc -e -in pic\_original.bmp -out pic\_cbc.bin -K 00112233445566778889aabbccddeeff -iv 0102030405060708

4.复制原始图片头信息到加密文件：

dd if=pic\_original.bmp of=pic\_ecb.bmp bs=1 count=54 conv=notrunc

dd if=pic\_original.bmp of=pic\_cbc.bmp bs=1 count=54 conv=notrunc

dd if=pic\_ecb.bin of=pic\_ecb.bmp bs=1 skip=54 seek=54

dd if=pic\_cbc.bin of=pic\_cbc.bmp bs=1 skip=54 seek=54

1. 观察结果

ECB模式加密图片：

加密后的图片仍能看出原始图片的大致轮廓和结构

高对比度区域（如边缘）在加密图片中仍可见类似模式

这是因为ECB模式独立加密每个块，相同明文块产生相同密文块

CBC模式加密图片：

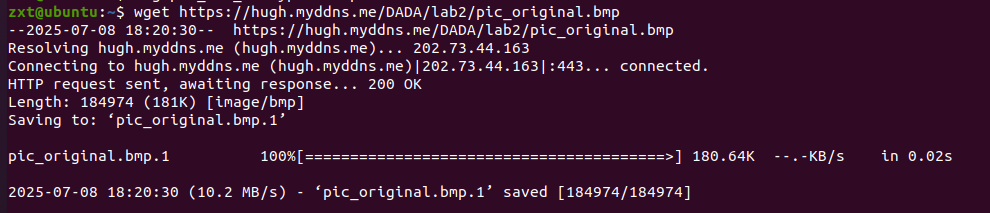
加密后的图片显示为完全随机的噪声

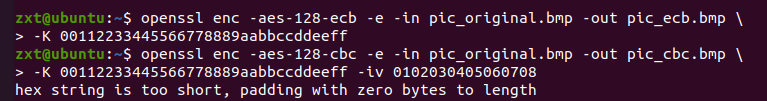
无法辨认任何原始图片的特征或轮廓

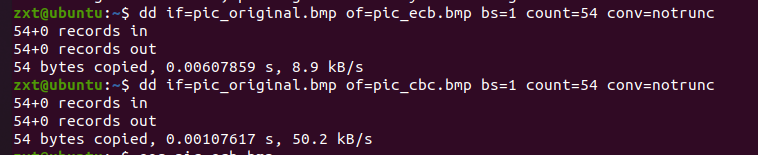
这是因为CBC模式通过IV和链式加密引入了依赖性，相同明文块会产生不同密文块

6.结论

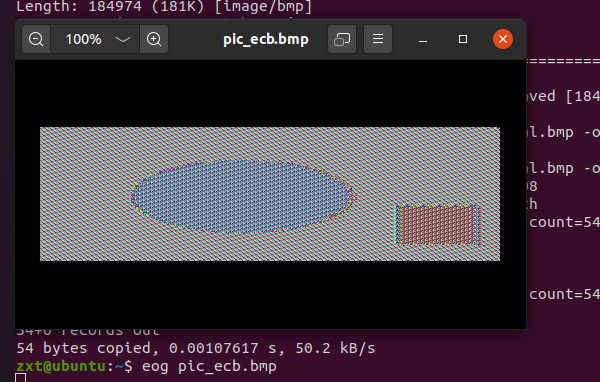
ECB模式不适合加密图像等包含大量重复模式的数据，因为它会泄露结构信息。而CBC模式通过引入初始向量和块间依赖性，提供了更好的安全性，完全隐藏了原始图像的特征。

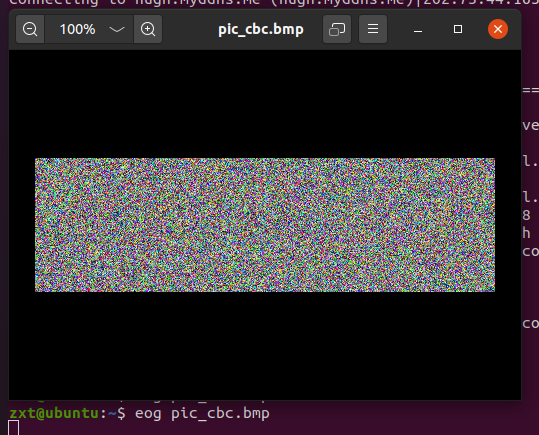






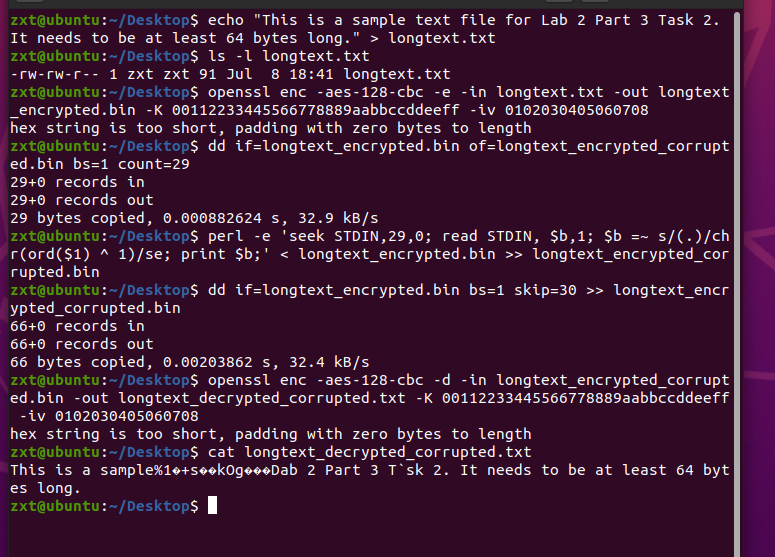




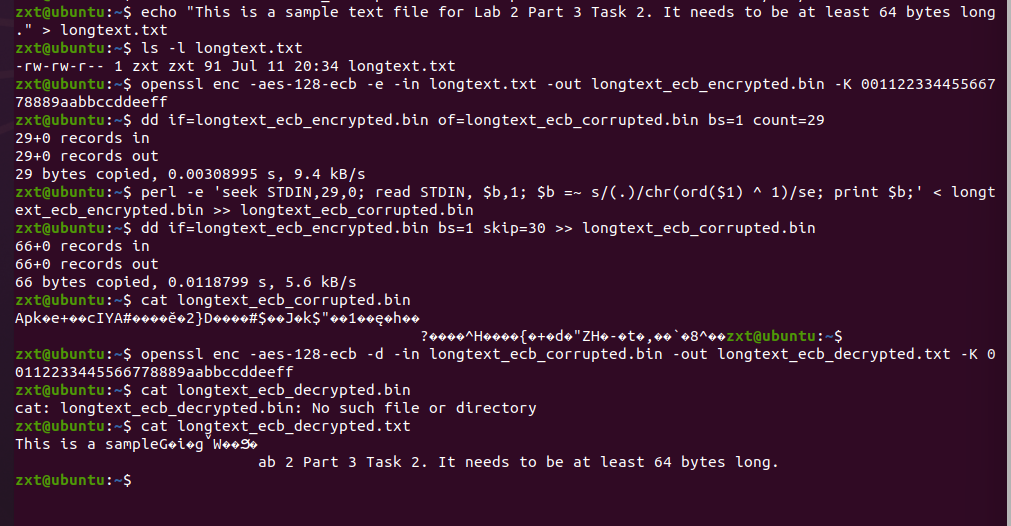


**Task2**

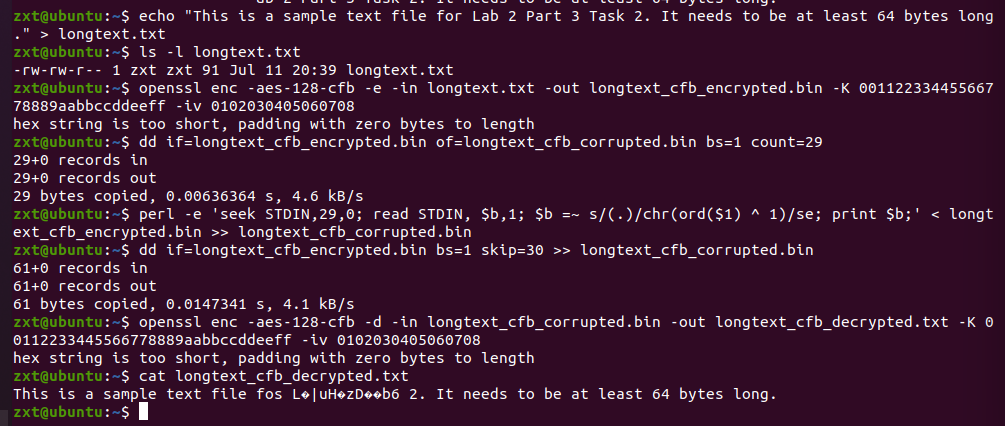
1.Cbc:



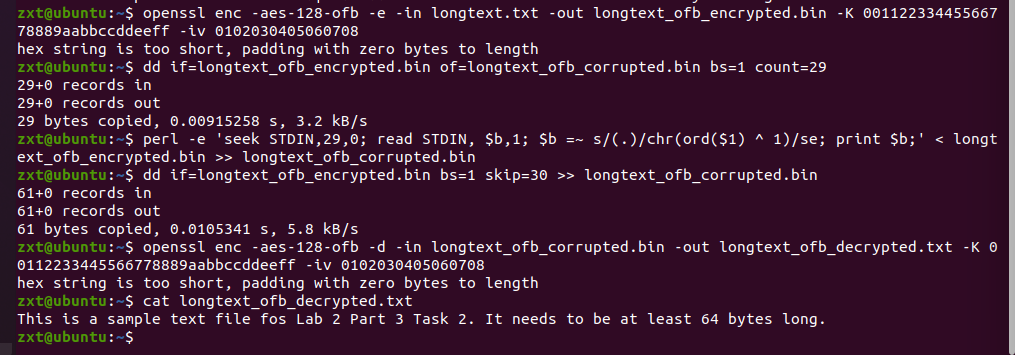
1. Ecb



1. Cfb



1. ofb：



**1. 明文被成功恢复了多少？**

ECB：

第30字节位于第2个块（偏移量16-31），仅第2块完全损坏，其余块正常解密。

恢复的明文：第1块（0-15字节）、第3块及后续块（32字节以后）完全正确。

CBC：

第2块（16-31字节）：完全损坏（解密后乱码）。

第3块（32-47字节）：仅第32字节（对应第30字节的位置）损坏，其余正确。

恢复的明文：第1块（0-15字节）完全正确；第3块部分正确（仅1字节错误）。

CFB：

第30字节：单字节损坏。

后续所有字节（31字节以后）：完全损坏（错误传播至文件结束）。

恢复的明文：仅前29字节正确，后续全部乱码。

OFB：

仅第30字节损坏，其余字节完全正确。

恢复的明文：仅1字节错误，其他全部正确。

**2. 错误（损坏的位）影响了解密明文的哪些部分？**

ECB：仅影响第2块（16-31字节），其他块无影响。

CBC：

第2块（16-31字节）完全乱码。

第3块（32-47字节）中第32字节错误（与第30字节位置对齐）。

CFB：

第30字节错误 + 后续所有字节（31字节以后）乱码。

OFB：仅第30字节错误，无传播。

**3. 解密过程是否因填充错误而完全失败？**

ECB/CFB/OFB：

不会失败（ECB无依赖，CFB/OFB为流模式无填充）。

CBC：

若损坏的字节位于最后一个块的填充区（如文件末尾），可能触发 bad decrypt，但解密仍会输出（部分数据可能损坏）。

本例中第30字节不在最后一块，不会因填充失败，但会因块依赖导致错误传播。