COMP2700 实验室 9 - 加密模式

在本实验室中，我们将了解不同的加密模式，即如何使用分块密码对大于密码分块大小的数据进行加密，以及不当使用加密模式产生的安全问题。

本实验室的部分练习选自 Paar & Pelzl 的 "理解密码学 "一书
(第 5 章）。

除了 pycryptodome 库，我们还将使用 **openssl** 命令执行 AES 加密/解密，并使用 Linux **dd** 命令执行位块复制。这两个工具在任何标准的 Linux 安装中都可以使用，所以如果你愿意，也可以使用自己的 Linux 安装。

要在 ECB 模式下使用 AES 128 位加密，请使用以下方法：

```
openssl enc -aes-128-ecb -nosalt -in input-file -out output-file
```

将提示您输入用于生成 AES 密钥的密码。该命令在 ECB 模式下使用 128 位 AES 加密输入文件，并在输出文件中输出结果。解密命令几乎相同，只是增加了"-d "选项：

```
openssl enc -d -aes-128-ecb -nosalt -in input-file -out output-file
```

要使用 CBC 模式加密，只需将 -aes-128-ecb 替换为 -aes-128-cbc 即可。

注意：在 CBC 模式下，选项 -nosalt 会导致每次加密都使用相同的 IV。切勿在实际操作中使用该选项，因为它会引发各种安全问题。本实验室使用该选项仅为测试目的。

我们可以使用 dd 命令复制字节块。我们将举例说明该命令的语法：

dd if=input.bin **of=output**.bin bs=16 skip=2 count=2

该命令从输入文件（*if=input.bin*）复制 2 个字节块（在选项 *count=2 中*指定），每个字节块由 16 个字节组成（*bs=16*），跳过头两个字节块（*skip=2*），并将结果输出到 output.bin 文件（*of=output.bin*）。你可以根据自己的特殊需要调整参数。在本实验中，我们将只使用 128 位（16 字节）的数据块大小，因此将使用 bs=16 选项。

练习 1 和练习 3 需要额外的文件。这些文件可以从本实验室的 Wattle 页面下载（参见文件 lab-9-files.zip），或者直接从以下网址 "wget "下载：

wget http://users.cecs.anu.edu.au/~tiu/comp2700/lab9_files.zip

在练习 3 中，我们将使用 python 中的 PyCryptodome 库，特别是其中的 strxor 模块，对字节数组执行 XOR 操作。

练习 1.在本题中，我们将对本教程课程网站中提供的著名的 Linux 版 Tux 企鹅*徽标*（文件 *Tux.ppm*）进行加密。我们将使用一种名为 PPM 的文件格式：参见

https://en.wikipedia.org/wiki/Netpbm_format#PPM_example

PPM 格式有两种：一种是使用 ASCII 字符对每个像素的颜色（RGB 代码）进行编码（P3 类型），另一种是使用二进制表示对颜色进行编码（P6 类型）。我们将使用后者。P6 类型 PPM 格式的页眉由三行文本组成：第一行包含 "P6 "字样；第二行包含两个数字，以像素为单位指定图像的宽度和高度；第三行是每种颜色的最大值（范围为 1 - 65535）。第三行的典型值为 255 或 65535。例如，本实验室中的 Tux.ppm 文件有如下标题

```
P6
265 314
255
```

也就是说，这是一个 P6 类型的 PPM 图像文件，大小为 265x314 像素，每种颜色用小于或等于 255 的值表示。文件头后面的内容就是像素的位图，每个像素用像素的三个值（RGB 颜色）表示。如果每种颜色的最大值小于或等于 255，则每种颜色值由一个字节表示。如果大于 255，则用 2 个字节表示。

a) 在 ECB 模式下使用 AES-128 位加密 Tux.ppm 的图像内容（像素图，不包括标题）。如何只提取像素（不含标题）？使用什么命令?
b) 尝试将 a) 中的加密内容作为图像查看。如何查看？你观察到了什么?
c) 现在使用 128 位 AES，以 CBC 模式重复上述操作。您在输出图像中看到了什么?

练习 2.本题旨在说明 ECB 模式用于加密长度超过一个区块的信息时的危险性，因为信息的含义可能取决于多个区块。

考虑以下（假设的）银行间资金转账协议。协议的核心部分使用以下报文格式对交易进行编码：

AAA:XXXXXXXXXX:BBB:YYYYYYYY:CCC:<amount to be transferred> 该报

文包含 6 个以冒号分隔的字段：

- AAA：汇款银行的三个字母代码。
- XXXXXXXXX：发送账户的 11 位数账号。
- BBB：收款银行的三个字母代码。

- YYYYYYYY：接收账户的 11 位数账号。
- CCC：转账货币的三个字母代码。
- 最后一个字段是一个长度可变的字段，包含要转账的金额。例如，以下

信息

CBA:11122233344:ANZ:01234567890:AUD:100

其中包含一项从 CBA 的 11122233344 账户向 ANZ 的 01234567890 账户转账 100 澳元的指示。

假设银行在 ECB 模式下使用 AES 128 加密来加密交易。

a) 使用 openssl 工具加密以下两个交易。将每笔交易保存到一个文件中，然后在 ECB 模式下使用 AES 对文件进行加密。您可以使用任何密码来生成加密密钥。

   CBA:82934681003:NAB:99203848881:AUD:120
   CBA:82934681003:ANZ:45200943921:AUD:3500

b) 仅使用 a) 中的加密报文，为下面的交易创建另一条加密报文：

   CBA:82934681003:NAB:99203848881:AUD:3500

   (注意：不允许使用 a) 中的加密密钥来执行此操作，即设想攻击者设法截获了 a) 中的密文，并希望伪造另一个传输。请使用 dd 命令来帮助您处理密码文本）。

练习 3.本题给您三个文件：ofb1.txt、ofb1.enc 和 ofb2.enc。(这些文件可以从本实验室的 Wattle 页面下载）。

b1.enc文件是用某个密钥k和某个IV值I以OFB模式加密b1.txt后得到的。b2.enc文件是用相同的密钥以 OFB 模式加密另一个文件后得到的。
和 IV 值 I。请说明如何在不知道密钥和 IV 值的情况下解密 ofb2.enc。

Python 库 Crypto.Util.strxor

要解决练习 3，我们将使用 Pycryptodome 库（特别是 Crypto.Util.strxor 模块）对两个字节字符串执行 XOR。

要以字节数组形式读取二进制文件，可以使用 "打开 "函数打开文件，并使用读取函数读入字节数组。完成后，别忘了使用 "close "函数关闭文件。

下面是一个读取 "file1"（二进制文件）内容的 python 交互会话示例文件）转换成字节数组 arr1。

```
>>> f=open("file1","rb")
>>> arr1=f.read()
>>> arr1
b'abcd'
>>> f.close()
>>>
```

函数 open("file1", "rb") 将 "file1 "作为二进制文件打开供读取。在这种情况下，字节数组 arr1 的值由字符串 "abcd "中每个字符的 ASCII 码组成。(在 Python 中，可以将字符串视为字节数组。这样的数组通常以前缀 `b` 显示，例如 `b'abcd'`）。

要将一个字节数组与另一个字节数组进行 XOR 运算，我们需要使用 strxor 函数。下面是一个 XOR 操作的示例；继续前面的示例（因此 arr1 的值是字节数组 `b'abcd'`）。

```
>>> arr2=b'\x00\x00\x00\x00'
>>> arr2
b'\x00\x00\x00\x00'
>>> from Crypto.Util.strxor import *
>>> strxor(arr1,arr2)
b'abcd'
>>> arr3=b'\xff\xff\xff\xff'
>>> strxor(arr1,arr3)
b'\x9e\x9d\x9c\x9b'
>>> arr4=b'\x20\x20\x20\x20'
>>> strxor(arr1,arr4)
b'ABCD'
>>>
```

在本例中，我们首先声明一个变量 arr2，并将其初始化为 4 个 0 字节；这里我们使用字节的十六进制值，每个字节表示为"\xHH"，其中 HH 是字节的十六进制值。

strxor 函数位于 Crypto.Util.strxor 模块中，使用前必须先导入该模块。我们在此展示了三个

---

strxor 的示例，分别位于 arr1 与 arr2、arr3 和 arr4 之间。请注意，arr4 是一个 4 字节数组，其中每个元素都包含空格""字符的字节值（\x20）。

练习 4.我们使用计数器模式 (CTR) 下的 AES 加密一个硬盘，其容量为 1 TiB（= 2^{40} 字节，约 1.1 TB）的容量。IV 的最大长度是多少？

扩展练习（可选）

练习 5.回想一下，AES 等块式密码的输入是一个固定大小的块，例如，在 AES 中是 128 比特或 16 字节。如果 AES 的输入不完全是 16 字节长，则需要添加填充字节。与块密码和某些加密模式（特别是 CBC 模式）一起使用的常用填充方案是 PKCS#7 填充[1] 方案。该方案的工作原

理如下：假设密码的块大小为 N 字节。如果输入m的最后一个区块长度为 (N - r) 字节，那么将r 的字节值添加到m中，直到m的最后一个区块长度为 N 字节。例如，如果 N = 16 字节（如 AES），m 是以下字节串（用 HEX 符号表示）：

000102030405060708090a0b0c0d0e0faabbccddeeff

---

[1] PKCS#7 的规范可在此处找到 https://www.ietf.org/rfc/rfc2315.txt

在这种情况下，信息 m 的长度为 22 字节，需要额外的 10 字节才能使其长度成为块大小（16 字节）的倍数。在这种情况下，$r = 10$（或用 HEX 表示为 0a），因此我们用 10 个字节填充字节串 m，每个字节都包含值 r（用 HEX 表示为 0a）：

000102030405060708090a0b0c0d0e0faabbccddeeff0a0a0a0a0a0a0a0a0a0a

请注意，如果报文 m 的长度已经是 N 的倍数，则会增加一个额外的数据块（该数据块的每个字节都包含 N 的字节值）。

在以下问题中，假设 N=16 个字节。

a. 假设以下数据块是 PKCS#7 填充的结果。
   0a11d34488220011f100aabb33010101
   填充前的原始信息是什么？
b. 下面是一个非常简单的填充方案。如果输入信息 m 不是 N 的倍数，则向 m 添加字节 00，直到其长度变成 N 的倍数。
   那么得到的填充报文就是
   00112233445566778899aabbcc000000
   这种天真无邪的填充方案会有什么潜在问题？

练习 6 (*)。在一家公司，所有在网络上发送的文件都是在 CBC 模式下使用 AES-128 自动加密的。使用固定密钥，IV 每天更改一次。网络加密以文件为基础，因此在每个文件的开头都使用 IV。

您设法获得了固定的 AES-128 密钥，但不知道最近的 IV。今天，您窃听到了两个不同的文件，一个内容不明，另一个已知是自动生成的临时文件，只包含 0xFF 值。请简要说明如何获得未知的 IV，以及如何确定未知文件的内容。

练习 7 (*)。在 OFB 模式下对 IV 保密不会使穷举密钥搜索变得更加复杂。请说明如何使用未知的 IV 进行暴力破解。对明文和密文有什么要求？

练习 8 (*)。想象一下，外星人并没有绑架地球人并对他们进行奇怪的实验，而是在地球上投放了一台特别适合 AES 密钥搜索的计算机。事实上，它的功能非常强大，我们可以在几天内搜索完 128、192 和 256 个密钥位。为外星人所需的明文-密文对数量提供指导，以便他们可

以合理地排除假密钥。