بسم الله الرحمن الرحیم

تمرین دوم درس مبانی امنیت شبکه های کامپیوتری

دکتر سعیدی

مهدی وجهی - ۸۱۰۱۰۱۵۵۸

سوال ۱

الف

به آسانی با یک حلقه شکل داده را ماتریسی می کنیم.

```
>>> key = '02020202020202020202020202020202'
>>> m_key = [[0 for j in range(4)] for i in range(4)]
>>> for i in range(16):
...     byte = key[i*2:(i+1)*2]
...     m_key[i%4][i//4] = byte
...
>>> print(np.array(m_key))
[['02' '02' '02' '02']
 ['02' '02' '02' '02']
 ['02' '02' '02' '02']
 ['02' '02' '02' '02']]
>>>
>>> plain_text = '0F0E0D0C0B0A09080706050403020100'
>>> m_plain_text = [[0 for j in range(4)] for i in range(4)]
>>> for i in range(16):
...     byte = plain_text[i*2:(i+1)*2]
...     m_plain_text[i%4][i//4] = byte
...
>>> print(np.array(m_plain_text))
[['0F' '0B' '07' '03']
 ['0E' '0A' '06' '02']
 ['0D' '09' '05' '01']
 ['0C' '08' '04' '00']]
```

ب

حال کلید را با داده XOR می کنیم به صورت درایه به درایه.

```
>>> tmp = [[0 for j in range(4)] for i in range(4)]
>>> for i in range(4):
...     for j in range(4):
...             t = int(m_plain_text[i][j], 16)
...             k = int(m_key[i][j], 16)
...
...             tmp[i][j] = f'{t ^ k:02X}'
...
>>> print(np.array(tmp))
[['0D' '09' '05' '01']
```

```
 ['0C' '08' '04' '00']
 ['0F' '0B' '07' '03']
 ['0E' '0A' '06' '02']]
```

ج

سپس با s-box عملیات sub byte رو انجام می دیم.

```
>>> # https://github.com/rafael2903/AES-128-cipher/blob/main/AES.py#L11
>>> s_box = (
...     0x63, 0x7C, 0x77, 0x7B, 0xF2, 0x6B, 0x6F, 0xC5, 0x30, 0x01,
0x67, 0x2B, 0xFE, 0xD7, 0xAB, 0x76,
...     0xCA, 0x82, 0xC9, 0x7D, 0xFA, 0x59, 0x47, 0xF0, 0xAD, 0xD4,
0xA2, 0xAF, 0x9C, 0xA4, 0x72, 0xC0,
.....
...     0x8C, 0xA1, 0x89, 0x0D, 0xBF, 0xE6, 0x42, 0x68, 0x41, 0x99,
0x2D, 0x0F, 0xB0, 0x54, 0xBB, 0x16
... )
>>>
>>> for i in range(4):
...     for j in range(4):
...             tmp[i][j] = f'{s_box[int(tmp[i][j], 16)]:02X}'
...
>>> print(np.array(tmp))
[['D7' '01' '6B' '7C']
 ['FE' '30' 'F2' '63']
 ['76' '2B' 'C5' '7B']
 ['AB' '67' '6F' '77']]
```

د

سپس هر ردیف را طبق شماره آن شیفت می دهیم.

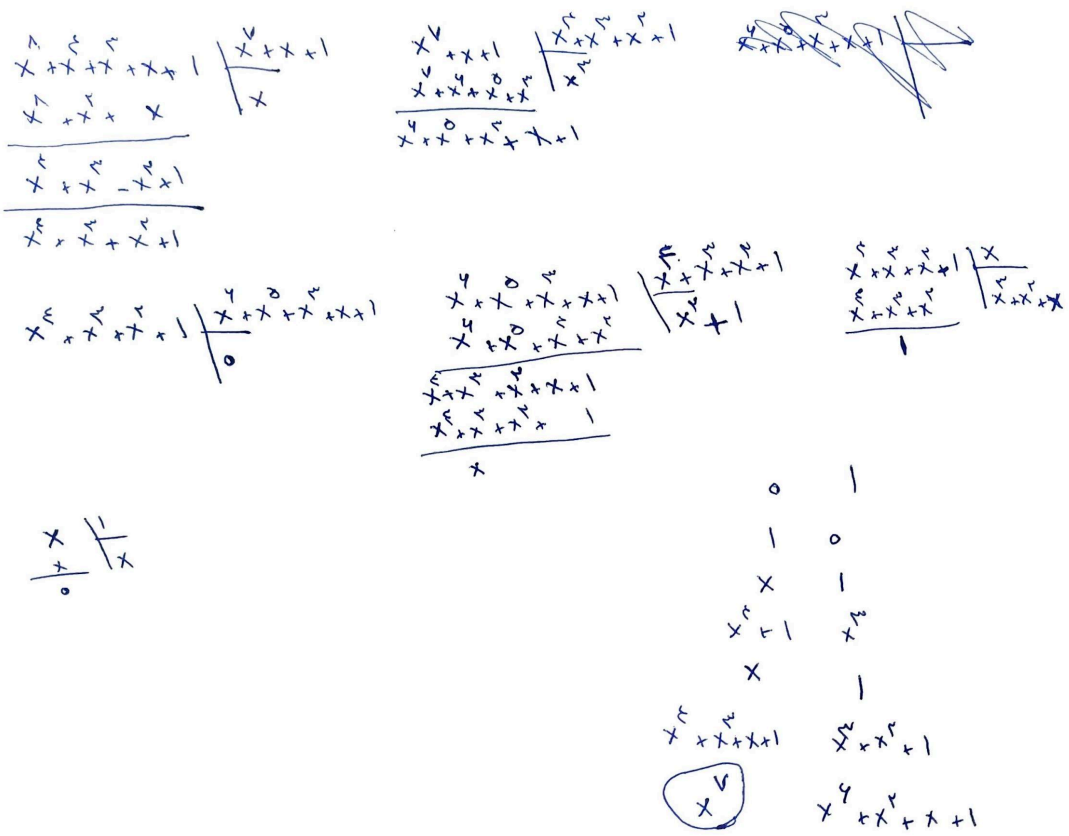
```
>>> for i in range(4):
...     for j in range(i):
...             tmp[i].append(tmp[i].pop(0))
...
>>> print(np.array(tmp))
[['D7' '01' '6B' '7C']
 ['30' 'F2' '63' 'FE']
 ['C5' '7B' '76' '2B']
 ['77' 'AB' '67' '6F']]
```

۵

سپس برای ترکیب ستون ها لازم است ضرب گالوانی رو تعریف کنیم و طبق ماتریس aes عملیات را انجام دهیم.

```
>>> # https://stackoverflow.com/a/70264791
>>> def mpy(x, y):
...     p = 0b100011011
...     m = 0
...     for i in range(8):
...         m = m << 1
...         if m & 0b100000000:
...             m = m ^ p
...         if y & 0b010000000:
...             m = m ^ x
...         y = y << 1
...     return m
...
>>> tmp_new = [[0 for j in range(4)] for i in range(4)]
>>> m_mc = [[2,3,1,1],
...         [1,2,3,1],
...         [1,1,2,3],
...         [3,1,1,2]]
>>>
>>> for i in range(4):
...     for j in range(4):
...             result = 0
...             for k in range(4):
...                     result = mpy(int(tmp[k][i],16), m_mc[j][k]) ^
result
...             tmp_new[j][i] = f'{result:02X}'
...
>>> print(np.array(tmp_new))
[['57' 'DF' '62' 'A5']
 ['94' 'D8' '50' '89']
 ['EF' 'E3' '4D' '65']
 ['79' 'C7' '66' '8F']]
```

سوال ۲



سوال ۳

الف

CTR مناسب است زیرا برای دسترسی تصادفی مناسب است و شباهت بلوک ها هم وجود ندارد. مشکل ECB تولید بلوک های تکراری (ناامنی) است اما مشکلی برای دسترسی تصادفی ندارد. CBC و CFB و تا حدی هم OFB در دسترسی تصادفی مشکل ساز است.

ب

CFB مناسب است زیرا درصورت از دست رفتن یک بسته امکان ادامه دادن وجود دارد. دلایل باقی موارد مانند الف است. نکته نا مناسب بودن CTR گم شدن بسته و عقب ماندن شمارنده است. (البته می توان در صورت عدم کدگشایی درست بسته با چندین عدد بعدی نیز آزمایش کرد و این موضوع را برای CTR حل کرد)

ج

CTR زیرا بلوک ها مستقل رمز میشود و اگر متن رمز شده به آن وارد شود کد گشایی می شود و برای متن ساده کد گذاری می شود. باقی موارد مانند حالت الف هست و به همان دلیل که تصادفی دسترسی ممکن نیست موازی سازی هم ممکن نیست.

د

CFB, CTR, OFB مناسب هستند زیرا ابتدا کلید مستقل از متن ورودی تولید می شود. با این روش ها کلید تولید می شود و به ترتیب با استریم متن XOR می شود.

سوال ۴

الف

ECB دیکود می کند اما بلوک c2 دو بار در خروجی ظاهر می شود. در CFB بلوک دوم c2 دیکود نمی شود ولی باقی متن بدون مشکل دیکود می شود.

ب

در CBC تنها بلوک اول خراب میشه. در CTR دیگر امکان کدگشایی نداریم.

ج

در OFB به دلیل زنجیر بود امکان کدگشایی از آنجا به بعد نیست مگر این که پیاده سازی هوشمند تر باشد و چند بلوک را بافر کند و با هرکدام تلاش کند کدگشایی کند. در CTR بلوک ها مستقل هستند و با جا به جایی دو بلوک تنها همان دو بلوک خراب کدگشایی می شوند و بقیه متن سالم است.

سوال ۵

برابری یا نابرابری دو متن رمز شده برابری یا نابرابری دو متن رمز نشده را نتیجه می دهد. زیرا کلید بین آنها یکسان بوده تنها عامل متفاوت IV بود که در اینجا یکسان است.

سوال ۶

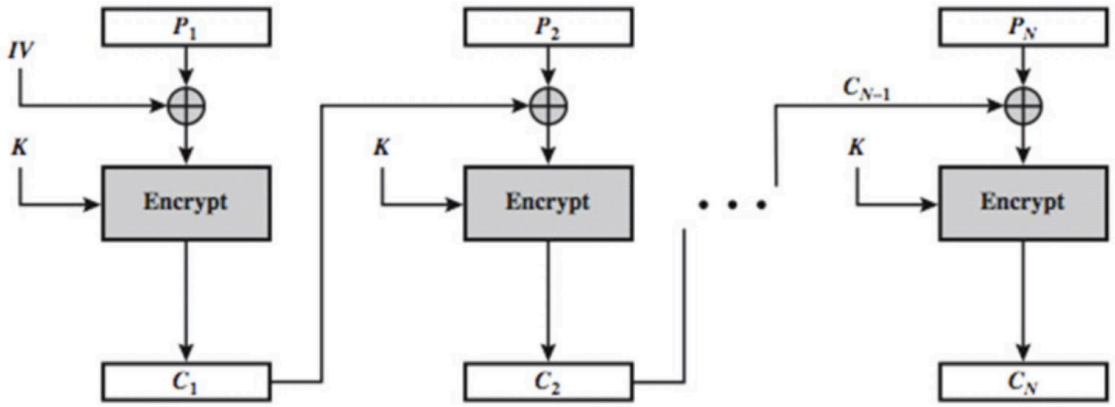
```python
# https://www.pycryptodome.org/src/examples#encrypt-data-with-aes
from Crypto.Cipher import AES
from Crypto.Random import get_random_bytes
from Crypto.Util import Padding

data = 'secret data to transmit'.encode()

print('-'*20, 'ECB', '-'*20)
print('plain text: ', data)

aes_key = get_random_bytes(16)

cipher = AES.new(aes_key, AES.MODE_ECB)
ciphertext = cipher.encrypt(Padding.pad(data, AES.block_size))
print(f'cipher text: {' '.join([hex(i) for i in ciphertext])}')

cipher = AES.new(aes_key, AES.MODE_ECB)
decrepted = Padding.unpad(cipher.decrypt(ciphertext), AES.block_size)
print('decrepted text: ', decrepted)

print('-'*20, 'CBC', '-'*20)
print('plain text: ', data)

aes_key = get_random_bytes(16)
aes_IV = get_random_bytes(16)
print(f'IV: {' '.join([hex(i) for i in aes_IV])}')

cipher = AES.new(aes_key, AES.MODE_CBC, aes_IV)
ciphertext = cipher.encrypt(Padding.pad(data, AES.block_size))
print(f'cipher text: {' '.join([hex(i) for i in ciphertext])}')

cipher = AES.new(aes_key, AES.MODE_CBC, aes_IV)
decrepted = Padding.unpad(cipher.decrypt(ciphertext), AES.block_size)
print('decrepted text: ', decrepted)
```

```
-------------------- ECB --------------------
plain text:  b'secret data to transmit'
cipher text: 0x80 0x6e 0x8f 0xa 0x75 0x94 0x3e 0x14 0x36 0x2e 0x5e 0x12
0x7f 0xa8 0xa1 0xf6 0x5b 0x9b 0xab 0x67 0xfb 0x16 0x7f 0x8a 0x73 0xd4
0x7a 0xb 0x85 0x2f 0xdd 0x7c
decrepted text:  b'secret data to transmit'
-------------------- CBC --------------------
```

```
plain text:  b'secret data to transmit'
IV: 0x4b 0x5e 0x50 0xca 0x8b 0x7d 0x3 0x82 0xaa 0x8b 0xa1 0x45 0x78 0xa3
0x27 0x51
cipher text: 0x8c 0x71 0x91 0xf1 0x66 0x7a 0x50 0xe1 0xf2 0xd6 0x8a 0xbd
0xf0 0x38 0x3b 0xc2 0xa6 0x5d 0xbe 0x77 0xaa 0x5 0x20 0xe0 0x37 0x50
0x2b 0x9c 0x61 0x22 0xfc 0xad
decrepted text:  b'secret data to transmit'
```

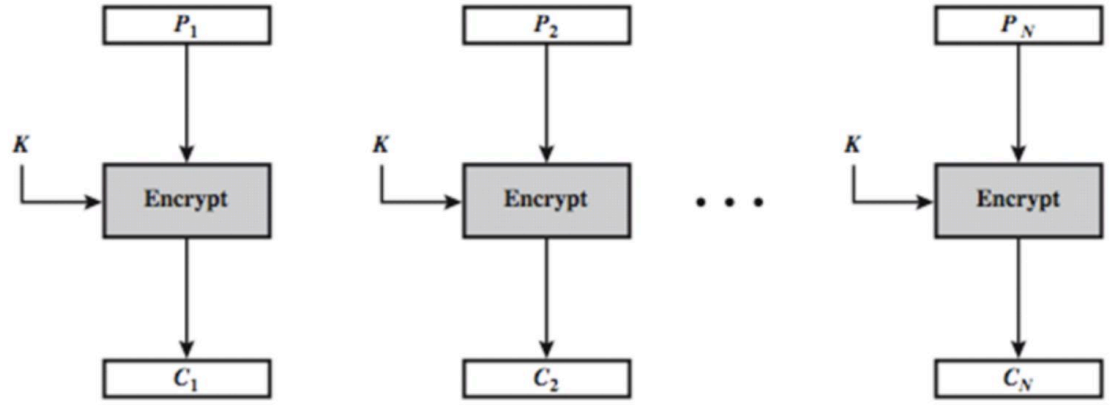
الف

همه ی نتیجه تغییر می کنه.



ب

بله. زیرا هیچ وابستگی ای به موقعیت بلوک ندارد این روش و با متن و کلید یکسان خروجی یکسان میده.



منابع

https://gemini.google.com/share/749c88520a09
https://gemini.google.com/share/c33067ee958f