数字电路xor加密实验

应承轩 201785071 电计1701

实验1 文本加密 in ACSII

formulation

给定"密钥"、"被加密文本"、"密文"，算法通过如下方式对文本进行加密：

密钥(tips)：任意文本都行，作为加密解密的载体用 被加密文本：可以是一段密码文本 密文：被加密后的文本（通常是乱码）

model

给定密钥 'ABCDEFGH' 给定被加密文本 'Z' 我们把每个被加密文本字符拆分成八位bit、分别xor进密钥的八位字符的每一位的最后一个bit位。（还有很多做法，我这么做比较冗余）

experiment

```
tips = ABSTRACT Relevance estimation is among the most important tasks in the rank- ing
of search results. Current relevance estimation methodologies mainly concentrate on text
matching between the query and Web documents, link analysis and user behavior models.

key = user:root,pwd:e3d8h12e

secret = @BRTS@BT!Seldw`nbe!esuhmauioo!hs `mnof thd mnru hlqnru`nu!u`sjr in!tid!ranj, hng
of rd`rbi!rdrtlts/ Ctsreot!sdlewaoce!dsuhmauhon mdthneologhdr mainmy!boocenusate!on!uexu
latbii

key_decode = user:root,pwd:e3d8h12e
```

通过对比，我们可以清楚的看到tip的字符要么不变要么变低一位，比如 ABSTRACT -> @BRTS@BT，这与我们的设计是一致的，只对目标最后一位进行xor，导致其在acsii编码表中的顺序-1或者+0，尽管如此，仍然损失了句意，在下面的图片实验中我们可以看到，对图片进行xor加密操作通常不会改变图片的语义。

另外，从实验结果可以看出文本语义变化很大，几乎不可读了，这是因为文本由于其离散性，xor之后都是变化很大（类似Bag-of-char模型，改变一个char，cos可能直接变成0了）。

code

```
tips = 'ABSTRACT Relevance estimation is among the most important tasks' \
    ' in the rank- ing of search results. Current relevance estimation' \
    ' methodologies mainly concentrate on text matching between the query' \
    ' and Web documents, link analysis and user behavior models.'
```

```
print('tips = {}'.format(tips))

key = 'user:root,pwd:e3d8h12e'
print('key = {}'.format(key))

secret = ''

# experiment1: 对tip中每个字符的最后一位进行xor

# encoder
pointer = 0
for char in key:
    for i in range(8):
        # get base value of key
        bit_of_key = (ord(char) >> i) % 2

        # get ord of tips
        int_of_tips = ord(tips[pointer])

        secret += chr(int_of_tips ^ bit_of_key)
        pointer += 1
print('secret = {}'.format(secret))

# decoder
pointer = 0
key_decode = ''
for start in range(0, len(secret), 8):
    t = 0
    for i in range(8):
        # get last bit
        last_bit_of_secret = ord(secret[start + i]) % 2
        last_bit_of_tip = ord(tips[start + i]) % 2

        # decode by xor
        bit_of_key = last_bit_of_secret ^ last_bit_of_tip
        base = (1 << i)
        t += bit_of_key * base

    # encode int to acsii char
    key_decode += chr(t)
print('key_decode = {}'.format(key_decode))
```

实验2 图片加密 in RGB by Xor

formulation

给定"密钥图片"、"被加密图片"、"目标图片"，算法通过如下方式对文本进行加密：

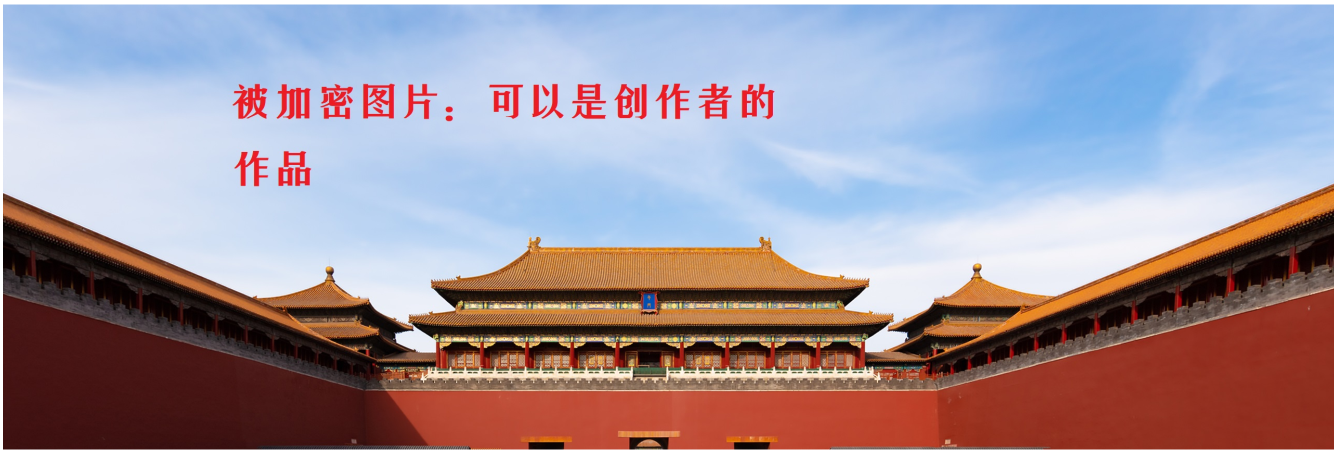
密钥图片：一个小于被加密图片的图片，可以用它证明图片原创者，或者xxxx目的 被加密图片：可以是创作者的作品 目标图片：被加密后的图片（通常是变化较小）

model

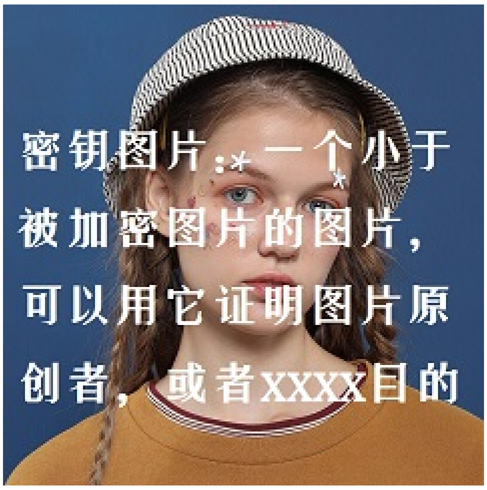
给定密钥图片 给定被加密图片 我们把每个密钥图片的RGB三通道拆成8位bit，分别xor进被加密图片的8个像素，然后得到目标图片。

experiment

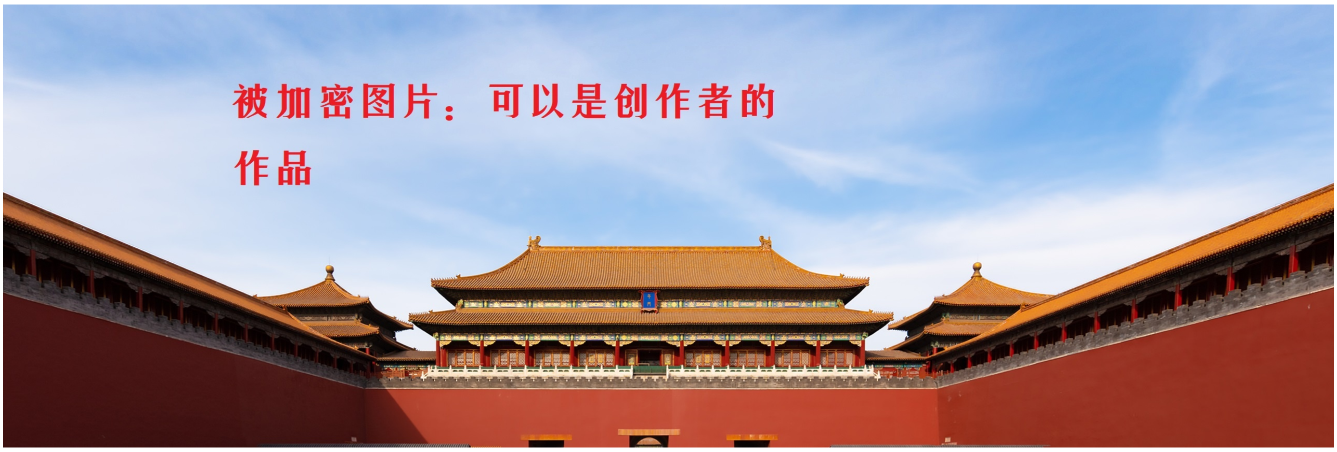
密钥图片:



被加密图片:

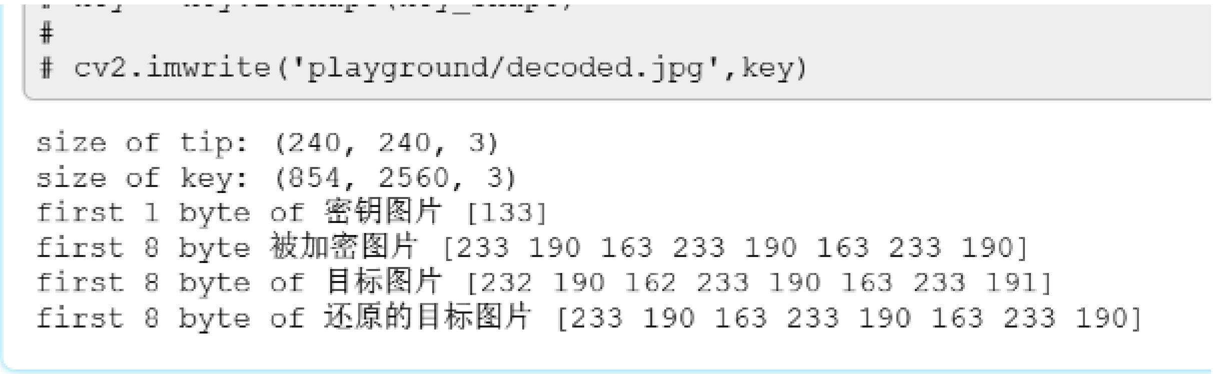


目标图片:



可以看出，没啥变化，并且改变的像素值并不影响对图片的理解（和文本xor很不一样）！

执行过程



```
#
# cv2.imwrite('playground/decoded.jpg',key)

size of tip: (240, 240, 3)
size of key: (854, 2560, 3)
first 1 byte of 密钥图片 [133]
first 8 byte 被加密图片 [233 190 163 233 190 163 233 190]
first 8 byte of 目标图片 [232 190 162 233 190 163 233 191]
first 8 byte of 还原的目标图片 [233 190 163 233 190 163 233 190]
```

可以看出，同字节加密类似的，目标图片和被加密图片差值位-1或者+0。

code

```
import cv2
tip = cv2.imread('playground/tip.jpg')
key = cv2.imread('playground/gg.jpg')

tip_shape = tip.shape
key_shape = key.shape

print('size of tip: {}'.format(tip_shape))
print('size of key: {}'.format(key_shape))

tip = tip.reshape((-1))
key = key.reshape((-1))
print('first 1 byte of 密钥图片',tip[:1])
print('first 8 byte 被加密图片',key[:8])

# encode
pointer = 0
for i in range(tip.shape[0]):
    int_of_pixel = tip[i]

    for i in range(8):
        # get base value of key
        bit_of_tip = (int_of_pixel >> i) % 2

        key[pointer] = key[pointer] ^ bit_of_tip
        pointer += 1

print('first 8 byte of 目标图片',key[:8])
# key = key.reshape(key_shape)
#
# cv2.imwrite('playground/encoded.jpg',key)


# decode
pointer = 0
```

```python
for i in range(tip.shape[0]):
    int_of_pixel = tip[i]

    for i in range(8):
        # get base value of key
        bit_of_tip = (int_of_pixel >> i) % 2

        key[pointer] = key[pointer] ^ bit_of_tip
        pointer += 1

print('first 8 byte of 还原的目标图片',key[:8])

# key = key.reshape(key_shape)
#
# cv2.imwrite('playground/decoded.jpg',key)
```