تکلیف عملی اول

مجتبی ملائی
۴۰۱۳۱۳۸۳

رسول صالحی
۴۰۱۲۵۹۳۳

۱ بارگذاری کتابخانه ها

[1]:
```
from Crypto.Hash import SHA256
from Crypto.Cipher import AES
from Crypto.Util.Padding import  pad,unpad
from PIL import Image
import numpy as np
```

۲ هش کردن

در این قسمت شماره های دانشجویی را توسط SHA-256 هش می‌کنیم. برای این کار یک آبجکت SHA256 می‌سازیم که دیتای ورودی آن شماره دانشجویی است. برای هش کردن لازم است که داده به صورت بایت باشد. بنابراین با استفاده از `encode()` به بایت تبدیل شده است. همچنین برای تبدیل دوباره به رشته از `hexdigest()` استفاده می‌کنیم که خروجی آن یک رشته باینری ۳۲ بایتی(۲۵۶بیتی) است و به صورت یک رشته هگزادسیمال ۶۴ حرفی نشان داده شده است.

[2]:
```
data = '40131383-40125933'
sha256_hash = SHA256.new(data=data.encode()).hexdigest()
sha256_hash
```

[2]:
```
'd54fa0c07db7f94eac27a4745a1a545806c501e7fa6b1b8b471b67a8aeb12ca5'
```

۳ ساخت کلید و IV

ابتدا ۳۲ حرف ابتدایی آن یعنی نصف ۲۵۶ بیت را برای کلید و سپس ۳۲ حرف دوم یعنی ۱۲۸ بیت بعدی را برای IV استفاده می‌کنیم. همچنین دوباره آنها را به بایت تبدیل می‌کنیم تا بتوانیم از آن برای رمز نگاری استفاده کنیم.

[3]:
```
key =  bytes.fromhex(sha256_hash[:int(265/8)])
iv =  bytes.fromhex(sha256_hash[int(256/8):])
```

۴ آماده سازی عکس

برای رمز کردن عکس نیاز داریم تا آن را به صورت بایت تبدیل کنیم. برای این کار ابتدا عکس را لود می‌کنیم و سپس پیکسل های عکس را به صورت RGB استخراج کرده و در غالب یک آرایه سه بعدی numpy ذخیره می‌کنیم. سپس ابعداد آرایه را برای فرم دهی مجدد عکس ذخیره می‌کنیم. سپس پیکسل ها را به صورت بایت ذخیره می‌کنیم. چون اندازه بایت های نهایی مضربی از ۱۲۸ بیت است، نیازی به پدینگ وجود ندارد.

[4]:
```
img = Image.open("image.png").convert("RGB")
pixels = np.array(img)
original_shape = pixels.shape
data = pixels.tobytes()
print(original_shape)
img
```

```
(224, 224, 3)
```

[4]:

۵ رمز کردن با ECB

برای رمز کردن عکس با استفاده از AES، یک آبجکت از AES با مود ECB می‌سازیم و سپس با استفاده از متد `encypt()` عکس را رمز می‌کنیم.

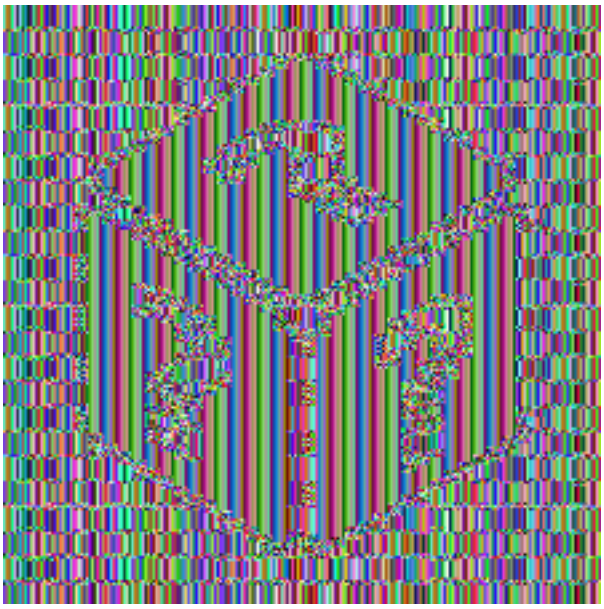
[5]:
```
ecb_cipher = AES.new(key= key,mode= AES.MODE_ECB)
data_ecb_enc =  ecb_cipher.encrypt(data)
```

۶ بررسی عکس رمز شده در مود ECB

بایت های رمز شده را دوباره به صورت آرایه سه بعدی RGB تبدیل می‌کنیم و آن را نمایش می‌دهیم. همانطور که می‌بینید شکل کلی عکس و الگوی آن قابل مشاهده است. در جاهایی که رنگ یکسانی داشتند، الگوی یکسانی شکل گرفته است.

[6]:
```
ecb_encrypted_array = np.frombuffer(data_ecb_enc, dtype=np.uint8)
ecb_encrypted_array = ecb_encrypted_array.reshape(original_shape)
ecb_encrypted_img = Image.fromarray(ecb_encrypted_array, mode="RGB")
ecb_encrypted_img.save("image_ecb_encrypted.png")
ecb_encrypted_img
```

[6]:



۷ رمز کردن با CBC

همانند قبل، فقط از مود CBC و IV استفاده شده است.

[7]:
```
cbc_cipher = AES.new(key=key,mode=AES.MODE_CBC,iv=iv)
data_cbc_enc =  cbc_cipher.encrypt(data)
```

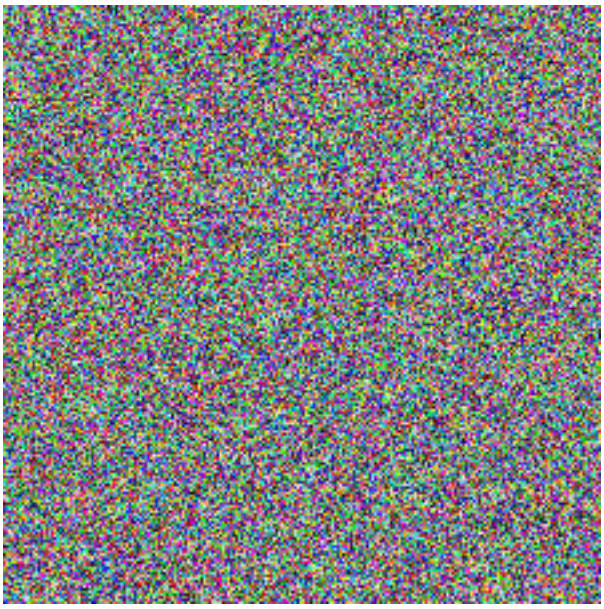
۸ بررسی عکس رمز شده در مود CBC

همانطور که مشاهده می‌شود، هیچ الگویی از عکس اصلی قابل مشاهده نیست.

[8]:
```
cbc_encrypted_array = np.frombuffer(data_cbc_enc, dtype=np.uint8).
 ↪reshape(original_shape)
cbc_encrypted_img = Image.fromarray(cbc_encrypted_array, mode="RGB")
```

```
cbc_encrypted_img.save("image_cbc_encrypted.png")
cbc_encrypted_img
```

[8]:



۹ رمزگشایی CBC

با استفاده از یک آبجکت AES، کلید و IV دیتا را دوباره رمزگشایی می‌کنیم.

[9]:
```
data_cbc_dec =  AES.new(key=key,mode=AES.MODE_CBC,iv=iv).decrypt(data_cbc_enc)
```

۱۰ تطابق عکس رمزگشایی شده با عکس اصلی

داده رمزگشایی شده را همانند حالت قبل نمایش می‌دهیم. خروجی نشان می دهد همان عکس است.

[10]:
```
cbc_decrypted_array = np.frombuffer(data_cbc_dec, dtype=np.uint8).
↪reshape(original_shape)
cbc_decrypted_img = Image.fromarray(cbc_decrypted_array, mode="RGB")
cbc_decrypted_img.save("image_cbc_decrypted.png")
cbc_decrypted_img
```

[10]:

برای اطمینان از تطابق کامل، از SHA256 استفاده می‌کنیم. روش هش نیز به ما اطمینان میدهد این دو عکس یکسان هستند.

[11]:
```
print('Match!') if SHA256.new(data=data).digest() == SHA256.
↪new(data=data_cbc_dec).digest() else print("not match!")
```

```
Match!
```

۱۱ تحلیل

به دلیل اینکه در ECB هر بلاک مستقل از بلاک دیگر رمز می‌شود و به مقدار دیگری مانند IV نیز ربطی ندارد، رمز شده یک بلاک یکسان همواره یک مقدار ثابت می‌ماند. بنابراین هر ۱۶ بایتی که یکسان باشند، به یک مقدار یکسان رمز خواهند شد. این باعث می‌شود که نقاطی که رنگ یسکانی داشته باشند، در عکس رمز شده نیز یکسان خواهند شد. در CBC هر بلاک روی بلاک های بعدی تاثیر می‌گذارد. بنابراین بلاک ها مستقل از یکدیگر رمز نمی‌شوند. در رمزنگاری عکس نباید از AES در مود ECB استفاده شود. چون باعث میشود الگو هایی از عکس ها که با بقیه عکس متفاوت هستند مانند نوشته ها فاش شوند. بنابراین باید از مود های دیگر استفاده شود.

۱۲ سوالات

۱. حالت Electronic Codebook (ECB) به این صورت عمل می‌کند که هر بلوک ۱۲۸ بیتی از داده‌های ورودی را به‌صورت مستقل رمزنگاری می‌کند. این ویژگی باعث می‌شود که اگر دو بلوک یکسان در ورودی وجود داشته باشد، خروجی رمزنگاری‌شده‌ی آنها نیز یکسان خواهد بود.

در مورد تصاویر، این خاصیت باعث می‌شود که ساختار کلی تصویر حفظ شود؛ زیرا نواحی مشابه در تصویر (مانند پس‌زمینه‌های یکنواخت) به بلوک‌های رمزنگاری‌شده‌ی یکسان تبدیل می‌شوند. در نتیجه، الگویی از تصویر اصلی در خروجی رمزنگاری‌شده قابل مشاهده است و این مسأله به افشا شدن اطلاعات تصویر و تحلیل رمز حمله‌گران کمک می‌کند.

۲. در حالت Cipher Block Chaining (CBC)، هر بلوک ورودی قبل از رمزنگاری با خروجی بلوک قبلی XOR می‌شود، و برای رمزنگاری اولین بلوک، از یک مقدار Initialization Vector (IV) تصادفی استفاده می‌شود.

تغییر IV باعث می‌شود که مقدار اولیه برای اولین بلوک تغییر کند و در نتیجه کل فرآیند رمزنگاری متفاوت باشد. این خاصیت موجب می‌شود که حتی اگر پیام‌های یکسانی دو بار رمزنگاری شوند، خروجی‌های متفاوتی تولید شوند، که امنیت را افزایش داده و مانع از تشخیص الگوها در متن رمزنگاری‌شده می‌شود.