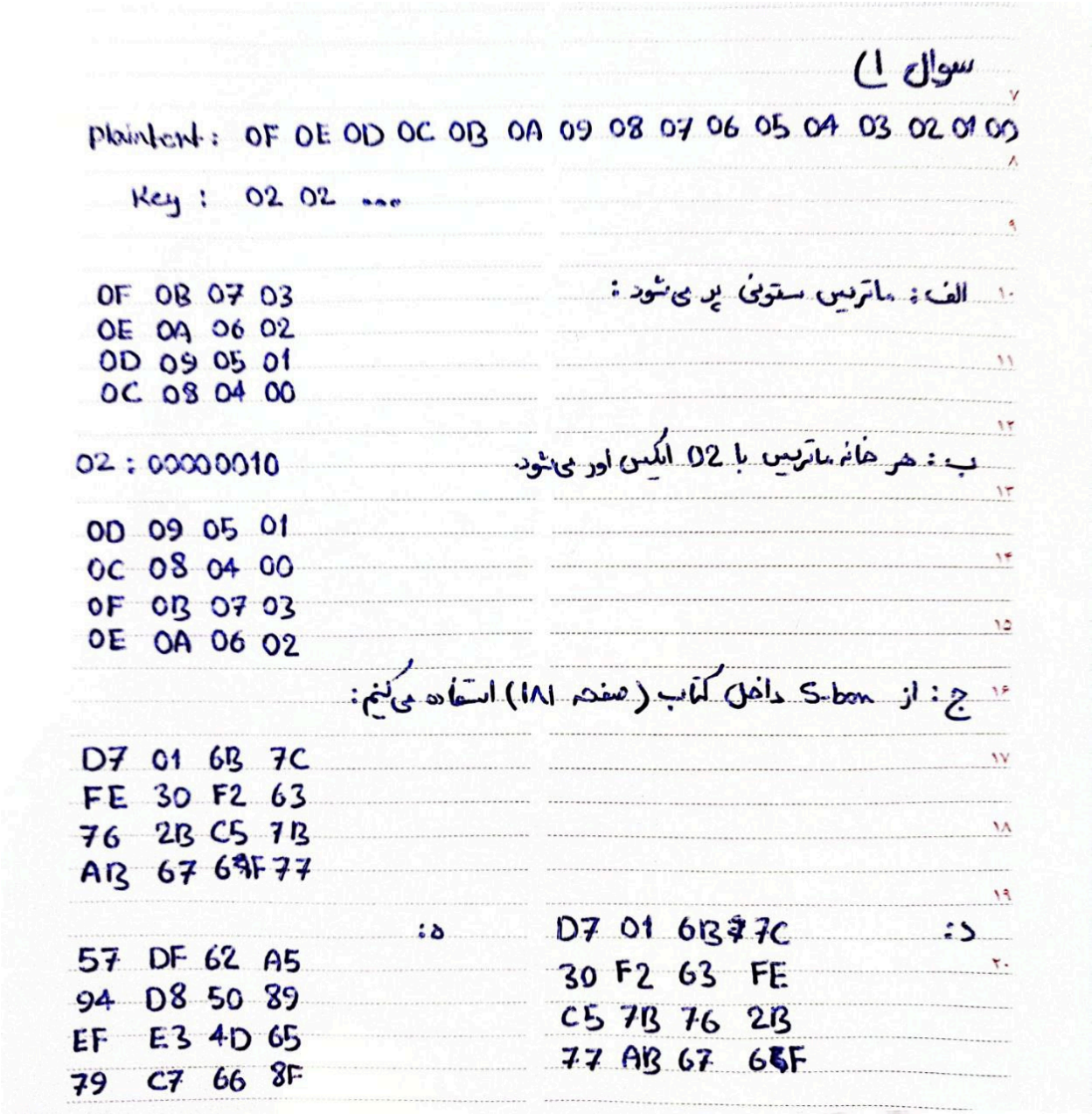
تمرین دوم امنیت شبکه - پاییز 1404

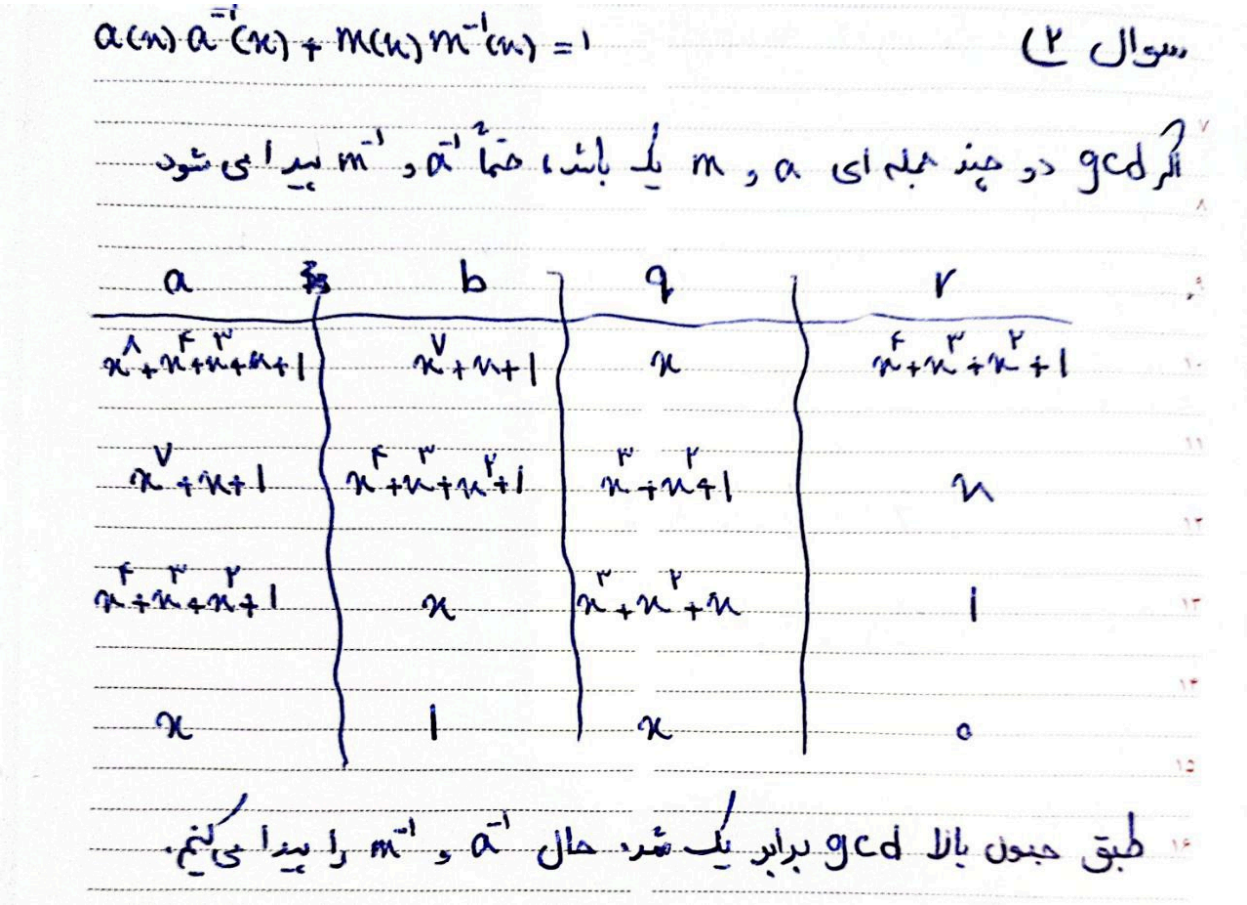
محمد مهدی صمدی - 810101465

سوال اول:

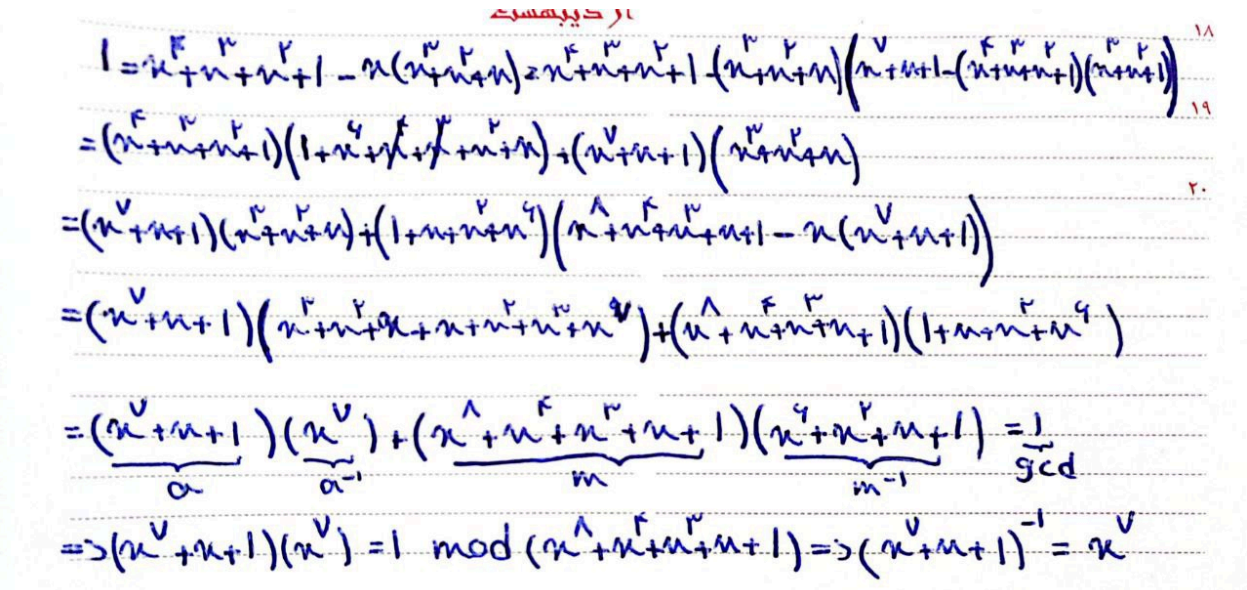


سوال دوم:

ابتدا gcd دو عبارت را پیدا می‌کنیم.



حال که شرایط مدنظر برقرار است معکوس ضربی آن‌ها را پیدا می‌کنیم.



سوالات سوم:

الف: رندوم اکسس خواسته شده به معنای این است که decrypt (همان خواندن از دیتابیس) یک بخش نیاز به این عمل روی بخش‌های قبلی نداشته باشد. پس CBC و CFB گزینه مناسبی نیستند. ECB هم به علت لو دادن الگو در دیتا (که قطعا در دیتابیس موجود است) مناسب نمی‌باشد. استفاده از هر کدام OFB یا CTR خوب است البته که CTR بهتر می‌باشد.

ب: CBC مناسب نیست زیرا ارور در یک بلاک به ادامه ویدئو منتقل می‌شود. CFB از جهت بازیابی بعد از ارور مشکل دارد. ECB از لحاظ معیار confidentiality مشکل دارد. همچنین mode هایی که واحد block دارند برای استریم ویدیو شاید مناسب نباشند و delay بندازند. پس مجدد مانند بخش الف OFB یا CTR مناسب هستند.

ج: CBC شرایط پراسس موازی را ندارد. همچنین CFB و OFB نیز عملکرد سریعی در این زمینه ندارند. میان ECB و CTR، چون سرعت مهم است ECB را انتخاب می‌کنیم (overhead به مراتب کمتری دارد).

د: برای پرهیز از padding باید ECB و CBC خط می‌خورند. سه حالت دیگر (CTR / OFB / CFB) شرایط مسئله را برقرار می‌کنند. اما اگر به دنبال موازی‌سازی یا شاخصه‌های دیگری باشیم شاید یکی از آنان مثل CTR تنها کاندید بشود.

سوال چهارم:

الف: ECB به صورت deterministic خروجی می‌دهد پس به ازای دو ورودی یکسان، دو خروجی یکسان می‌دهد و می‌تواند الگوی متن را لو دهد. همچنین ECB هیچ تغییری روی بلاک‌های دیگر نمی‌دهد. اما CFB به علت ماهیت sequential که دارد، خروجی برابری برای دو ورودی یکسان نخواهد داشت و همچنین نسبت به حالتی که تکرار نداریم خروجی از آن بلاک به بعد تغییر می‌کند.

ب: در CTR تغییر یک بیت در initial vector باعث خرابی کل خروجی می‌شود اما در CBC این خرابی فقط بلاک خروجی اول را خراب می‌کند و بقیه‌شان درست می‌مانند.

ج: در OFB خواهیم داشت که آن دو بلاک خروجی خراب می‌شوند (لزوماً مانند ورودی جابه‌جا نمی‌شوند) اما بلاک‌های بعدی بدون مشکل خواهند ماند. در CTR آن دو بلاک خروجی جابه‌جا می‌شوند و مانند حالت قبل مشکلی در بلاک‌های بعدی نخواهد بود.

سوال پنجم:

می‌دانیم روابط $C_i = E_k(P_i \oplus IV)$ و $P_i = D_k(C_i) \oplus IV$ برقرار هستند. با دانستن رابطه دوم داریم:

$$P_i^A = D_k(C_i^A) \oplus IV \,,\; P_i^B = D_k(C_i^B) \oplus IV$$

$$\rightarrow \quad P_i^A \oplus P_i^B = D_k(C_i^A) \oplus D_k(C_i^B)$$

از روی رابطه گفته شده نمی‌توان $D_k(C_i^A)$ یا $D_k(C_i^B)$ را به دست آورد. اما مهاجم می‌تواند در صورتی که دو خروجی یکسان باشند استنتاج کند که دو ورودی نیز یکسان بودند. برای همین هیچ‌وقت نباید یک زوج IV و کلید یکسان را بیش از یک بار استفاده کنیم.

سوال ششم:

کد استفاده شده در این بخش:

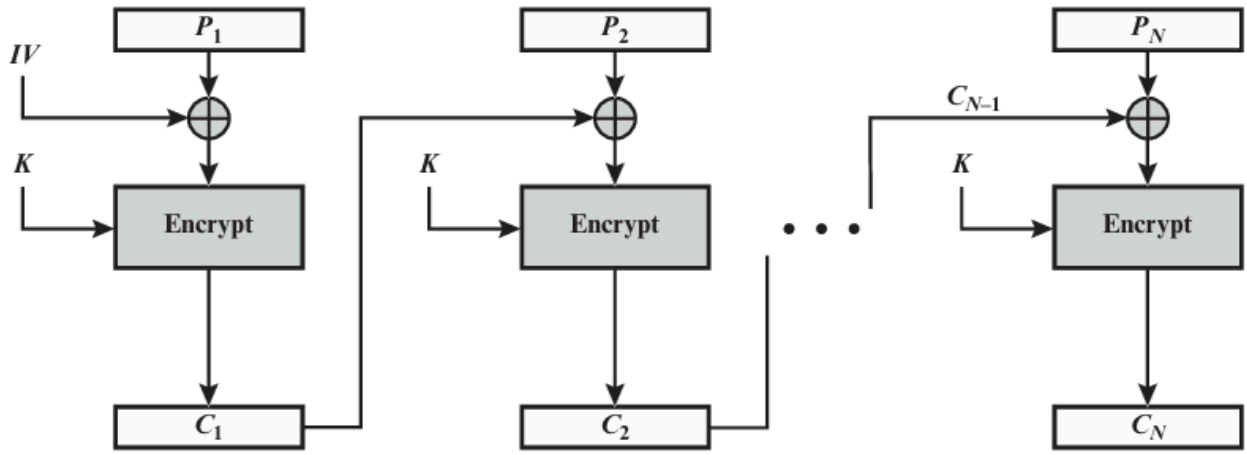
```
def to_hex(b: bytes) -> str:
    return binascii.hexlify(b).decode()

def pad(data: bytes) -> bytes:
    pad_len = 16 - len(data) % 16
    return data + bytes([pad_len] * pad_len)

def unpad(data: bytes) -> bytes:
    pad_len = data[-1]
    return data[:-pad_len]

def encrypt_ecb(key: bytes, plaintext: bytes) -> bytes:
    cipher = AES.new(key, AES.MODE_ECB)
    return cipher.encrypt(pad(plaintext))

def decrypt_ecb(key: bytes, ciphertext: bytes) -> bytes:
    cipher = AES.new(key, AES.MODE_ECB)
    return unpad(cipher.decrypt(ciphertext))

def encrypt_cbc(key: bytes, plaintext: bytes, iv: bytes) -> bytes:
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return cipher.encrypt(pad(plaintext))

def decrypt_cbc(key: bytes, ciphertext: bytes, iv: bytes) -> bytes:
    cipher = AES.new(key, AES.MODE_CBC, iv)
    return unpad(cipher.decrypt(ciphertext))

key = bytes.fromhex('02020202020202020202020202020202')
plaintext_str = "spread your wings and fly away, fly away, far away."
plaintext_bytes = plaintext_str.encode()

ciphertext_ecb = encrypt_ecb(key, plaintext_bytes)
decrypted_ecb = decrypt_ecb(key, ciphertext_ecb)
```

```
iv = get_random_bytes(16)
ciphertext_cbc = encrypt_cbc(key, plaintext_bytes, iv)
decrypted_cbc = decrypt_cbc(key, ciphertext_cbc, iv)

print("Simulating AES algorithm...")
print("plaintext:", plaintext_str)
print("IV:", to_hex(iv))

print("ciphertext (ECB):", to_hex(ciphertext_ecb))
print("ciphertext (CBC):", to_hex(ciphertext_cbc))

print("decrypted (ECB):", decrypted_ecb.decode())
print("decrypted (CBC):", decrypted_cbc.decode())

if plaintext_bytes == decrypted_ecb == decrypted_cbc:
    print("\nresult of an encryption followed by a decryption using the same
key is equal to the original message.")
else:
    print("\nfailed.")
```

خروجی برنامه نوشته شده:

```
PS E:\uni\daryl\Network Security\HWs\HW2> py .\Q6.py
Simulating AES algorithm...
plaintext: spread your wings and fly away, fly away, far away.
IV: b4245e5ca10253cb7086f629c0686893
ciphertext (ECB): 45359608ed326fc09b84a6d37370732960353480e2a0568cae8d28c99aefcdeba
59d44c8c3889ec44c9b2e61ed7f7932dbee31467d64a3cb4972ee5b085aa325
ciphertext (CBC): a66701661a16257e24b033113d9d6985523c41947c03035bacabed2b3b3dd6ba3
353b905c1445ae10d01317815f45353956e963ab96a18c86a2a1c27f46a8003
decrypted (ECB): spread your wings and fly away, fly away, far away.
decrypted (CBC): spread your wings and fly away, fly away, far away.

result of an encryption followed by a decryption using the same key is equal to the
 original message.
```

الف: اگر initial vector عوض شود، طبق تصویر زیر (از کتاب مرجع) ورودی باکس encrypt اولیه عوض شده و در نتیجه تغییر تا آخرین بلاک propagate می‌شود. پس تمام ciphertext عوض می‌شود.



ب:

این تصویر عملکرد ECB را نشان می‌دهد که طبق آن همه‌ی بلاک‌ها مستقل از هم کد می‌شوند. پس دو بلاک با ورودی یکسان به خروجی یکسان می‌انجامند.