به نام خدا

تمرین سوم امنیت داده و شبکه

نیم‌سال دوم ۱۴۰۴-۱۴۰۳

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

موضوع **امنیت شبکه**

موعد تحویل **سه‌شنبه ۱۶ اردیبهشت ۱۴۰۴**

طراحی تمرین توسط **رضا سعیدی - متین آقامیرکریمی - مبین آقامیرکریمی**

۱. در زیر دو متن رمز شده فارسی (بدون فاصله و علائم نگارشی) در اختیار داریم. می‌دانیم فقط یکی از آنها با روش ویجنر [1] رمزشده و دیگری تصادفی است.

راهنمایی

روش رمزنگاری ویجنر یکی از روش‌های رمزنگاری کلاسیک چندالفبایی است. این روش با استفاده از یک کلیدواژه برای شیفت دادن حروف متن اصلی به حروف رمز شده عمل می‌کند. (جزئیات بیشتر)

چچیعشعشعرحبفسژریقغبذنشاگض

شخسحححجیدارمشالهعثایکگططضچ

آ. ابتدا ضریب انطباق [2] را برای هر کدام حساب کرده و مشخص کنید کدام یک با احتمال بالاتری متن رمزشده فارسی می‌باشد. میدانیم طول کلید استفاده شده برابر ۴ و متن آشکار شامل عبارت "بخش دوم" است. با این اطلاعات متن انتخابی خود را رمزگشایی کنید. (پاسخ خود را کامل شرح دهید، صرفا رسیدن به نتیجه با کمک کدنویسی مدنظر نمی باشد)

ب. متن زیر با کمک رمزنگاری سزار رمز شده است. با کمک نتیجه به دست آمده بخش آ، متن زیر را رمزگشایی کنید.

فعپنپضعسعضسقزشظستو

۲. در جدول زیر، S-Box شماره ۸ مورد استفاده در الگوریتم رمزنگاری DES را مشاهده می‌کنید. یکی از معیارهای امنیت در DES این است که S-Box ها باید به صورت غیرخطی باشند. در این بخش می‌خواهیم مقدار S_1 که جدول آن در شکل نمایش داده شده است، را برای جفت ورودی‌های مختلف محاسبه کنیم.

شما باید بررسی کنید عبارت $(S_1(x_1) \oplus S_1(x_2)) \neq S_1(x_1 \oplus x_2)$ که شرط غیرخطی بودن است، برای هر جفت ورودی صحیح است یا خیر.

راهنمایی

بیت اول و آخر شماره سطر و چهار بیت وسط شماره ستون را مشخص می‌کنند.

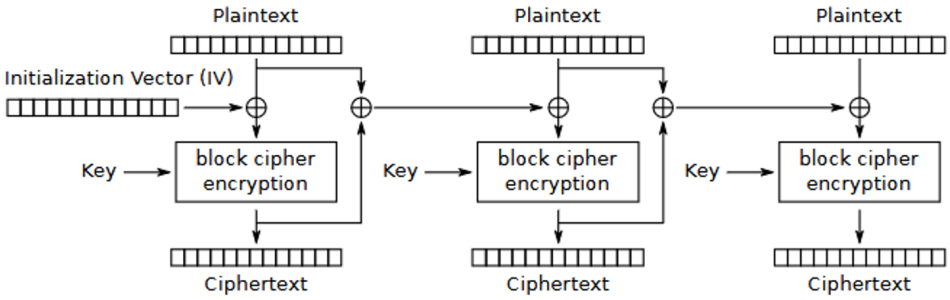
S_8	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	13	02	08	04	06	15	11	01	10	09	03	14	05	00	12	07
1	01	15	13	08	10	03	07	04	12	05	06	11	00	14	09	02
2	07	11	04	01	09	12	14	02	00	06	10	13	15	03	05	08
3	02	01	14	07	04	10	08	13	15	12	09	00	03	05	06	11

آ. $x_1 = 000000$ ، $x_2 = 000001$

ب. $x_1 = 111111$ ، $x_2 = 100000$

[1] Vigenere
[2] index of coincidence

۳. شکل زیر فرایند رمزکردن در حالت کاری انتشار زنجیره رمز قالبی [۳] را نشان می‌دهد.



آ. تاثیر بروز خطا در انتقال یک قالب متن رمزشده بر فرایند رمزگشایی در این حالت کاری را با CBC مقایسه کنید و مزیت PCBC را بیان کنید.

ب. نشان دهید در این حالت کاری اگر دو قالب رمزشده متوالی جابجا شوند، رمزگشایی ادامه پیام (بعد از این دو قالب) به‌درستی انجام می‌شود.

۴. پایگاه‌داده یک شرکت دارای n سطر است و سطر i ام با کلید k_i رمز شده است . برای آن که مجبور به نگهداری n کلید نباشیم، کلید رمزگذاری طبق فرایند زیر با تابع چکیده‌ساز H تولید می‌شود:

$$(1)\ 1 \le i \le n \rightarrow k_i = p_i \oplus q_i$$
$$(2)\ 1 < i \le n \rightarrow p_i = H(p_{i-1})$$
$$(3)\ 1 \le i < n \rightarrow q_i = H(q_{i+1})$$

در نتیجه تنها کافی است دو مقدار تصادفی q_n ، p_1 را تولید و به طور امن نگهداری کنیم. زیرا تمام کلیدها از این دو مقدار قابل تولید هستند.

آ. می‌خواهیم به یک کارمند دسترسی به سطرهای موجود در بازه $[a, b]$ را بدهیم. برای این کار پیشنهاد شده است تنها دو مقدار (p_a, k_b) در اختیار کارمند قرار گیرد. روش رمزگشایی سطر $j \in [a, b]$ام توسط کارمند را بیان کنید و توضیح دهید چرا این کارمند نمی‌تواند سطرهای خارج از این بازه را رمزگشایی کند.

ب. نشان دهید در روش پیشنهاد شده ممکن است دو کارمند، در صورت همکاری با یکدیگر، بتوانند سطرهایی که هیچ یک از آن دو اجازه دسترسی ندارند را رمزگشایی کنند.

ج. چه مقادیری به کارمندان بدهیم که دسترسی غیر مجاز بیان شده در قسمت قبل ممکن نباشد؟

[3] Propagating cipher block chaining (PCBC)

۵. در این سوال قصد داریم که به یک مقاله معروف در مورد حملات RSA اشاره کنیم.

این مقاله با عنوان Twenty Years of Attacks on the RSA Cryptosystem منتشر شده است. در مقاله نشان داده شده است که RSA در شرایط خاصی آسیب‌پذیر است و امکان شکستن آن وجود دارد. در این سوال، حمله Wiener مورد نظر قرار گرفته است. کد زیر پیاده‌سازی از الگوریتم RSA است:

```python
from Crypto.Util.number import getPrime, bytes_to_long, inverse
from random import getrandbits
from math import gcd

FLAG = b"Network_Security{XXXXXXXXXXXXXXXXXXXXXXXXXXXX}"

m = bytes_to_long(FLAG)

def get_huge_RSA():
    p = getPrime(1024)
    q = getPrime(1024)
    N = p*q
    phi = (p-1)*(q-1)
    while True:
        d = getrandbits(512)
        if (3*d) ** 4 > N and gcd(d,phi) == 1:
            e = inverse(d, phi)
            break
    return N,e

N, e = get_huge_RSA()
c = pow(m, e, N)

print(f'N = {hex(N)}')
print(f'e = {hex(e)}')
print(f'c = {hex(c)}')
```

راهنمایی

در حمله Wiener، کلید خصوصی کوچک می‌تواند با استفاده از کسرهای پیوسته بازیابی شود و امنیت RSA را به خطر بیندازد.

با اینکه تلاش بر این بوده که از وقوع حمله Wiener جلوگیری شود اما این سیستم همچنان امن نیست. ابتدا به طور کامل دلیل ناامنی در این شرایط خاص را توضیح دهید و سپس راهی برای استخراج FLAG پیدا کنید. (خروجی این کد در یک فایل txt به پیوست شده است.)

۶. عملیات رمزنگاری و رمزگشایی را با استفاده از الگوریتم RSA برای موارد زیر انجام دهید:

آ. $p = 3 ، q = 7 ، e = 5 ، M = 10$

ب. $p = 7 ، q = 17 ، e = 11 ، M = 11$

ج. $p = 17 ، q = 23 ، e = 9 ، M = 7$

۷. به پیوست تمرین یک کلید خصوصی RSA با فرمت PEM ارائه شده است. با استفاده از ابزار OpenSSL به سؤالات زیر پاسخ دهید:

آ. برای محافظت بهتر، این کلید خصوصی با کلمه عبور "network_sec_1404" محافظت می‌شود. این محافظت چگونه انجام میشود؟ به بیان دیگر چگونه اطمینان حاصل می‌شود تنها کسی که کلمه عبور را می‌داند، بتواند از کلید خصوصی استفاده کند؟

ب. مقدار $\varphi(n)$ را برای این کلید بدست آورید.

ج. به پیوست تمرین یک فایل رمزشده با قسمت عمومی این کلید ارائه شده است. آن را رمزگشایی کنید.

د. کلید عمومی را از فایل PEM استخراج کرده و سپس پیام آشکار بدست آمده در قسمت قبل را با استفاده از آن مجدد رمز کنید. آیا متن رمزشده بدست آمده با آنچه در ابتدا به شما ارائه شده بود یکی است؟ دلیل این امر چیست؟

ه. سعی کنید یک فایل بزرگ (مثلا یک تصویر) را با این کلید رمز کنید. خواهید دید که این امکان وجود ندارد. چه پارامتری در کلید باعث ایجاد این محدودیت می‌شود؟ برای رمزکردن فایلهای بزرگ چه روشی را پیشنهاد می‌کنید؟

۸. قطعه کد زیر یک پیاده سازی ناامن استفاده از روش CBC-MAC است.

```python
from Crypto.Cipher import AES

BLOCK_SIZE = 128 // 8

def enc_mac(k, m):
    # PKCS pad
    r = BLOCK_SIZE - len(m) % BLOCK_SIZE
    pad_size = r if r != 0 else BLOCK_SIZE
    m += pad_size.to_bytes(1, 'big') * pad_size

    # encrypt
    c = AES.new(mode=AES.MODE_OFB, key=k, iv=k).encrypt(m)

    # MAC
    t = AES.new(mode=AES.MODE_CBC, key=k,
    ↪  iv=c[:BLOCK_SIZE]).encrypt(m)[-BLOCK_SIZE:]

    return (c, t)
```

آ. دو متن رمزشده (c_1, t_1) و (c_2, t_2) با کلید k رمز و احراز صحت شده‌اند. هر دو را رمزگشایی کرده و اعتبار کد احراز صحت پیام‌ها را بررسی کنید.

$$k = \texttt{875faffbaeea63eb878613b98460f4d2}$$

$$(c_1, t_1) = (\texttt{d8b8239628a3f44c81e50cbd57aaac62586cdf1376c25fa8c23e8becf6be4688},\ \texttt{abb859c60dd1450bd789a40bc3638f4e})$$

$$(c_2, t_2) = (\texttt{f8a0238928a3fc4b9efa1aef03aaa62e4f668c0633dc21cdba4dafe3f9b14987},\ \texttt{b893a8d5032f5c004f11543626fc942e})$$

ب. متن رمزشده و کد احراز صحت پیام زیر را در نظر بگیرید. می‌دانیم متن آشکار **delete all keys** بوده و کد احراز صحت پیام آن معتبر است. بدون داشتن کلید، متن رمزشده پیام را طوری تغییر دهید که متن آشکار بدست آمده در سمت گیرنده برابر **everything is ok** شده و کد احراز صحت پیام همچنان معتبر باشد. دلیل این ناامنی را توضیح دهید.

$$(c, t) = (\texttt{fb7c5373f3713de7f41cee2ee49e09ef}, \texttt{33228cc0d41f4c94bfb7b2c47b5f69cd})$$