به نام خدا

# امنیت داده و شبکه

نیم‌سال دوم ۱۴۰۳-۱۴۰۲

دانشکده مهندسی کامپیوتر

دانشگاه صنعتی شریف

---

موضوع **رمزنگاری**

موعد تحویل **ساعت ۲۳:۵۹ شنبه ۲۲ اردیبهشت ۱۴۰۳**

طراحی تمرین توسط **امیرمحمد آقاپور**

# ۱ کلاسیک

یکی از انواع رمز جانشینیِ چندالفبایی، روش ویژنر [۱] است. در این روش هر حرف از متن آشکار به اندازه حرف متناظر با آن در کلید منتقل می‌شود. برای مطالعه بیشتر می‌توانید از این منبع استفاده کنید. عبارت زیر یک جمله فارسی (بدون فاصله و علائم نگارشی) رمز شده با جانشینی ویژنر است. می‌دانیم متن آشکار شامل عبارت **باران بهاری** بوده و اندازه کلید ۹ است. متن آشکار را پیدا کنید. [۲]

«قنفریفتفنیخحدارزجتتخمظیقتفضبفرذمعسفلگدثژ»

# ۲ در میان AESها

یکی از روش‌های اجرای حمله فراگیر [۳] در سیستم‌های رمزگذاری قالبی که از ترکیب چند مرحله رمزگذاری یا رمزگشایی ساخته شده‌اند، حمله ملاقات در میانه [۴] است. در این حمله برای کاهش پیچیدگی زمانی، از حافظه برای نگهداری مقادیر میانی در فرایند رمزگذاری استفاده می‌شود. برای مطالعه بیشتر می‌توانید از این منبع استفاده کنید. در هر بخش از این سؤال یک جفت متن آشکار و رمزشده در پایه ۱۶ [۵] به همراه کد مورد استفاده [۶] برای رمزگذاری ارائه شده است. کلید مورد استفاده در هر مرحله را بدست آورید. پیچیدگی زمانی و حافظه در بدترین حالت و زمان اجرای حمله خود را گزارش کنید.

**(آ)** $(p, c) = (\texttt{2f20cb8872c99b696e6461cb906c202f}, \texttt{fd25a141381dbaef0fafc20ce028d934})$

```python
from Crypto.Cipher import AES
from os import urandom # OS random generator

def aes_enc(p):
    k = urandom(2) + b'\0' * 14
    E = AES.new(mode=AES.MODE_ECB, key=k).encrypt
    c = E(bytes.fromhex(p))
    return c.hex()
```

**(ب)** $(p, c) = (\texttt{2f20cb8872c99b696e6461cb906c202f}, \texttt{e4714ee833977599b7ec0a8d83a62164})$

```python
def double_aes_enc(p):
    k1 = urandom(2) + b'\0' * 14
    k2 = urandom(2) + b'\0' * 14
    E1 = AES.new(mode=AES.MODE_ECB, key=k1).encrypt
    D2 = AES.new(mode=AES.MODE_ECB, key=k2).decrypt
    c = D2(E1(bytes.fromhex(p)))
    return c.hex()
```

[1] Vigenère

[۲] می‌توانید از ابزار برخط Cryptii برای انجام محاسبات خود استفاده کنید.

[3] Brute-force attack

[4] Meet-in-the-middle attack

[5] Hexadecimal

[۶] در کدهای این تمرین از کتابخانه PyCryptodome استفاده شده است که توصیه می‌شود برای پیاده‌سازی راه حل خود نیز از همین کتابخانه استفاده کنید.

**(پ)** $(p, c) = (\texttt{2f20cb8872c99b696e6461cb906c202f}, \texttt{a6addbf32d0c6c5c87e311d3a35f78d3})$

```
def triple_aes_enc(p):
    k1 = urandom(2) + b'\0' * 14
    k2 = urandom(2) + b'\0' * 14
    D1 = AES.new(mode=AES.MODE_ECB, key=k1).decrypt
    E2 = AES.new(mode=AES.MODE_ECB, key=k2).encrypt
    c = E2(D1(D1(bytes.fromhex(p))))
    return c.hex()
```

## ۳ RSA با OpenSSL

به پیوست تمرین یک کلید خصوصی RSA (`private.pem`) ارائه شده است. به کمک OpenSSL به سؤالات پاسخ دهید[۷]:

**(آ)** این کلید خصوصی با کلمه عبور DNS14022 محافظت می‌شود. این محافظت چگونه انجام شده است؟ 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.

**(پ)** به پیوست تمرین یک فایل رمزشده (`enc.bin`) با قسمت عمومی این کلید ارائه شده است. آن را رمزگشایی کنید.

**(ت)** پیام آشکار بدست آمده در قسمت قبل را دوباره با همین کلید رمز کنید. آیا متن رمزشده بدست آمده با آنچه در ابتدا به شما ارائه شده بود یکی است؟ دلیل این امر چیست؟

**(ث)** 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؟

## ۴ کلید بدشانس

در فرایند تولید دو کلید عمومی RSA زیر، به دلیل ضعف در مولد اعداد تصادفی، یک عامل اوّلِ مشترک استفاده شده است. با کمک این ضعف، کلید خصوصی هر دو را بدست آورید.

$$pk_1 = (n_1, e_1) = (882389665577830838482125131852013816279695311, 65537)$$
$$pk_2 = (n_2, e_2) = (726247788835915752041026275800104626981008161, 5)$$

## ۵ DH کوچک

پارامترهای عمومی و مقادیر انتخاب شده توسط دو طرف ارتباط در یک اجرای الگوریتم دیفی-هلمن[۸] در ادامه ارائه شده است. می‌دانیم یکی از طرفین مقدار خصوصی کوچکی انتخاب کرده است. کلید مشترک را بدست آورید.

$$q = 288918539521089348336793240678493497771$$
$$\alpha = 3$$
$$\alpha^{X_A} = 127823777105479486190202117586831854 25 \pmod q$$
$$\alpha^{X_B} = 183364455173249021598006044125891817111 \pmod q$$

[۷] می‌توانید از این ارهنما برای پاسخ به این سؤال استفاده کنید.

[8] Diffie–Hellman

## ۶ ضدّ نشت

می‌خواهیم رمز عبور کاربران را در پایگاه داده‌ای ذخیره کنیم که احتمال می‌دهیم در آینده نشت اطلاعات از آن رخ دهد. اگر $u$ نام کاربری، $p$ رمز عبور آشکار، $h$ تابع درهم‌ساز امن، $salt$ یک رشته تصادفی، $k$ کلید خصوصی سرور و $E_x(.)$ رمز بلوکی در حالت CTR با مقدار اولیه شمارنده صفر و کلید $x$ باشد؛ در هر حالت زیر ابتدا روش بررسی صحت رمز عبور را بیان کرده و سپس روش‌های پیشنهاد شده را از نظر امنیتی با ذکر دلیل مقایسه کنید.

**آ)** $(u, p)$ و $(u, h(p))$ و $(u, h(h(p)))$

**ب)** $(u, h(p))$ و $(u, salt, h(salt\|p))$

**پ)** $(u, salt, h(salt\|p))$ و $(u, salt, E_{salt}(p))$

**ت)** $(u, E_k(salt), h(salt) \oplus p)$ و $(u, h(salt), E_k(salt) \oplus p)$[۹]

## ۷ OFB-ENC+CBC-MAC

قطعه کد زیر یک پیاده‌سازی 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 اصالت شده‌اند. هر دو را رمزگشایی کرده و اعتبار کد اصالت‌سنجی هر یک را بررسی کنید.

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

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

$$(c_2, t_2) = (\texttt{dfb3319a23e6bf4d88b20cf342a9ac62447cc04770dd2cd2bc5b87e0fab24a84}, \texttt{b893a8d5032f5c004f11543626fc942e})$$

**ب)** متن رمزشده و کد اصالت‌سنجی زیر را در نظر بگیرید. می‌دانیم متن آشکار «`1$ to original_destination`» بوده و کد اصالت‌سنجی آن معتبر است. بدون داشتن کلید، متن رمزشده و کد اصالت‌سنجی را طوری تغییر دهید که متن آشکار بدست آمده در سمت گیرنده برابر «`99$ to attacker`» شده و قابل تشخیص نباشد.

$$(c, t) = (\texttt{ad7fa3468caf0b5c01ec7be9b583fa350d2ce39b8cd57ee26270235cd6598592}, \texttt{905f6d5d03e5269a52aa3e33b558e764})$$

[۹] فرض کنید برای انجام عملیات XOR، برای آنکه طول دو عملوند یکی شود، رشته کوتاهتر با صفر دنباله‌زنی (pad) شده است.