**پروژه نهایی**

**عنوان: Secure Instant Messaging**

**درس:** مبانی امنیت شبکه‌های کامپیوتری

**استاد:** مهسا سعیدی

**دستیاران آموزشی:** ابوالفضل اسلامی، علی عابدینی

**نیمسال دوم سال تحصیلی ۰۴-۱۴۰۳**

# بخش اول (پیاده‌سازی کلاینت‌چت رمزنگاری شده)

در بخش اول از شما خواسته شده است یک کلاینت چت رمزنگاری شده ایمن و کارآمد را با استفاده از الگوریتم *Double Ratchet* پیاده‌سازی کنید. این الگوریتم یک پروتکل محبوب برای راه‌اندازی ارتباط امن است که در سیستم‌های چت دنیای واقعی مانند Signal و WhatsApp استفاده می‌شود. در پیاده‌سازی خود، از انواع ابزارهای رمزنگاری که در کلاس بحث شده‌اند (از جمله Digital signatures، Public key encryption، Key exchange و Authenticated encryption) استفاده خواهید کرد. لازم است برای پیاده‌سازی بخش‌های مختلف این تمرین از کتابخانه *PyCryptodome* استفاده کنید؛ بر همین اساس کد اولیه‌ای در اختیار شما قرار خواهد گرفت که شامل یک قالب پایه است و می‌توانید آن را تکمیل کنید تا ویژگی‌های عملکردی و امنیتی توضیح‌داده‌شده در زیر را برآورده سازید.

## ۱. کلاینت چت همراه با End-to-End Encryption

### ۱-۱. جزئیات پیاده سازی

کلاینت چت شما از الگوریتم Double Ratchet برای فراهم کردن ارتباطات End-to-End Encryption با سایر کلاینت‌ها استفاده خواهد کرد. برای ارزیابی کلاینت پیام‌رسان شما، بررسی خواهیم کرد که آیا دو یا چند نسخه از کلاینت های پیاده‌سازی شده توسط شما می‌توانند به درستی با یکدیگر ارتباط برقرار کنند یا خیر.

بهترین راه برای درک الگوریتم Double Ratchet، مطالعه مستقیم مستندات آن است. بنابراین از شما می‌خواهیم بخش‌های ۱، ۲ و ۳ از مستند رسمی Signal را در این لینک بخوانید.
**پیاده‌سازی شما باید الگوریتم Double Ratchet را مطابق با بخش ۳ از مستند مذکور با در نظر گرفتن تغییرات و توضیحات زیر به درستی اجرا کند:**

- همان‌طور که در بخش ۲.۳ مستند Signal توضیح داده شده است، شما می‌توانید از HKDF برای به‌روزرسانی کلیدهای Diffie-Hellman استفاده کنید. روش استفاده صحیح از HKDF در بخش ۵.۲ مستند آمده است.

- HKDF یک تابع key derivation است که در فایل `lib.py` اضافه شده است. بخش ۵.۲ مستند نحوه استفاده از آن را در پیاده‌سازی شرح می‌دهد. برای استفاده از API ، کد `lib.py` را بخوانید.

- در `lib.py` دو تابع مرتبط با HMAC وجود دارد:
  - ☐ `hmac_to_aes_key` برای تولید کلیدهای رمزنگاری AES
  - ☐ `hmac_to_hmac_key` برای تولید کلیدهای HMAC بیشتر

  بخشی از کار شما این است که تصمیم بگیرید در هر مورد از کدام تابع استفاده شود تا الگوریتم Signal به درستی پیاده‌سازی شود.

- از جفت‌کلیدهای ElGamal برای تبادل کلید Diffie-Hellman استفاده کنید. به تابع `generate_eg` در `lib.py` مراجعه کنید.

- از AES-GCM به‌عنوان الگوریتم رمزنگاری متقارن برای رمزنگاری پیام‌ها استفاده کنید. (با استفاده از کلیدهای ارسال و دریافت که طبق بخش ۲.۴ مستند استخراج می‌شوند.

- ورودی AD برای توابع ratchetEncrypt و ratchetDecrypt در بخش ۳.۴ و ۳.۵ مستند Signal را نادیده بگیرید؛ توجه داشته باشید نیازی به احراز هویت header پیام‌ها نیست.

- هر کلاینت دارای یک جفت‌کلید اولیه ElGamal خواهد بود. این کلیدها برای استخراج root-key اولیه برای نشست‌های ارتباطی جدید به کار خواهند رفت.

- کلیدهای عمومی از طریق certificateهای ساده توزیع می‌شوند. هر کلاینت در هنگام مقداردهی اولیه، certificate مخصوص به خود را که حاوی کلید عمومی ElGamal آن است تولید می‌کند. فرض بر این است که یک نهاد مرکزی مورد اعتماد (مثلاً سرور مدیریت‌شده توسط توسعه‌دهندگان اپلیکیشن پیام‌رسان) وجود دارد که می‌تواند certificateهای تولید شده توسط کلاینت‌ها را به صورت امن دریافت کند. این نهاد مرکزی روی هر گواهی‌ای که دریافت می‌کند، امضای دیجیتال ایجاد می‌کند تا اعتبار هویت صاحب گواهی را تایید کرده و از

دستکاری آن توسط مهاجمان جلوگیری کند. سپس certificateهای امضاشده به کلاینت‌ها بازگردانده می‌شوند تا هر کلاینت بتواند کلید عمومی ElGamal سایر کلاینت‌های سیستم را داشته باشد.

- توجه داشته باشید که باید هر بار که با AES-GCM رمزنگاری می‌کنید، یک IV تصادفی جدید تولید کنید. می‌توانید از تابع `gen_random_salt` در `lib.py` برای تولید این IV استفاده کنید. توجه داشته باشید که پیاده‌سازی شما می‌تواند IVها را به صورت متنی (plaintext) در header پیام ذخیره کند.

- هزینه حافظه برای ذخیره کلید در الگوریتم شما باید O(1) و مستقل از تعداد پیام‌های ارسال‌شده باشد.

- برای بررسی صحت عملکرد DH Ratchet، باید هر پیام را با احتمال ۱۰ درصد با انجام یک مرحله DH Ratchet ارسال کنید تا تطبیق صحیح کلید در سمت مقابل ارزیابی شود.

- پیاده‌سازی شما باید قادر باشد پیام‌هایی را که ممکن است از دست بروند، با تاخیر برسند یا خارج از ترتیب تحویل داده شوند، به‌درستی مدیریت کند. نحوه‌ی پیاده‌سازی این قابلیت در بخش ۲.۶ از مستندات الگوریتم Double Ratchet سیگنال شرح داده شده است و نسخه‌ی کامل این الگوریتم که در بخش ۳ آمده نیز شامل پشتیبانی از این نوع پیام‌ها می‌باشد. برای مثال، فرض کنید دو پیام A و B خارج از ترتیب دریافت می‌شوند، به‌گونه‌ای که پیام B پیش از پیام A به دست گیرنده می‌رسد. پیاده‌سازی شما باید بتواند پیام B را بلافاصله پس از دریافت رمزگشایی کند، بدون اینکه منتظر رسیدن پیام A بماند، و همچنین باید امکان رمزگشایی پیام A را در صورت رسیدن آن فراهم کند. حداکثر تعداد پیام‌هایی که می‌توانند در یک chain از دست بروند و همچنان قابل بازیابی باشند، ۱۰ عدد در نظر بگیرید.

### ۲-۱. مدل تهدید

هدف الگوریتم Double Ratchet فراهم کردن *امنیت پیش‌رونده (Forward Secrecy)* است (یعنی افشای کلیدهای بلندمدت یا کلید نشست فعلی نباید باعث افشای پیام‌های گذشته شود.)

به‌طور خاص، فرض کنید یک مهاجم Man-in-the-Middle به نام «دارث» بین آلیس و باب قرار گرفته است. دارث تمام پیام‌های رمزنگاری‌شده‌ای را که بین آلیس و باب رد و بدل می‌شود مشاهده کرده و آن‌ها را در یک فضای ذخیره‌سازی دائمی ذخیره می‌کند. سپس در مقطعی، دستگاه آلیس به خطر می‌افتد و دارث به کلیدهای مخفی فعلی آلیس دسترسی پیدا می‌کند. (فرض بر این است که آلیس مطابق توصیه‌های مستندات سیگنال، کلیدهای مربوط به پیام‌های قدیمی را حذف کرده است.) **پیاده‌سازی شما باید تضمین کند که در این سناریو، دارث حتی با دسترسی کامل به کلیدهای فعلی آلیس، قادر به رمزگشایی پیام‌های گذشته‌ای که ذخیره کرده، نباشد.**

پس از افشای کلیدهای آلیس، دارث می‌تواند یک حمله فعال Man-in-the-Middle انجام دهد؛ به‌گونه‌ای که خودش را جای هر دو طرف جا زده و بتواند تمام ارتباطات بین آلیس و باب را رمزگشایی کند. با این حال، در مرحله بعد، آلیس موفق می‌شود یک پیام را به باب ارسال کند بدون اینکه دارث بتواند آن را رهگیری کند. **در این حالت، پیاده‌سازی شما باید تضمین کند که دارث دوباره تمام دسترسی خود را برای رمزگشایی پیام‌ها از دست بدهد.** این ویژگی را *بازیابی پس از نفوذ (Break-in Recovery)* می‌نامند.

**پیاده‌سازی کامل الگوریتم Double Ratchet طبق مستندات سیگنال برای تضمین این دو ویژگی کافی است.**

## ۲. توضیحات API

در این بخش توضیحاتی درباره توابعی که باید پیاده‌سازی کنید آمده است.

### ۲-۱. `messenger.generate_certificate(username)`

این متد باید کلاینت پیام‌رسان را برای ارتباط با سایر کلاینت‌ها مقداردهی اولیه کند. یک جفت کلید ElGamal لازم برای تبادل کلید تولید کنید. کلید عمومی باید در یک certificate قرار گیرد تا برای

سایر کلاینت‌ها ارسال شود. شما آزاد هستید ساختار دلخواهی برای شیء certificate طراحی کنید، به شرط آنکه فیلدی به نام "username" داشته باشد.

### ۲-۲. `messenger.receive_certificate(certificate, signature)`

این متد یک certificate را از کلاینت دیگر دریافت کرده و در وضعیت داخلی پیام‌رسان ذخیره می‌کند، به‌گونه‌ای که از این پس بتوان پیام به مالک آن certificate ارسال یا از او دریافت کرد. آرگومان دوم این متد امضای مرجع مورد اعتماد بر روی گواهی است؛ شما باید صحت این امضا را (با استفاده از کلید عمومی مرجع مورد اعتماد که در سازنده کلاس پیام‌رسان در اختیار شما قرار داده می‌شود) بررسی کنید تا مطمئن شوید گواهی توسط یک مهاجم دستکاری نشده است. اگر نشانه‌ای از دستکاری تشخیص داده شد، بلافاصله یک exception ایجاد کرده و اجرای برنامه را متوقف کنید.

### ۲-۳. `messenger.send_message(name, message)`

این متد یک پیام رمزنگاری‌شده برای کاربری با نام مشخص‌شده ارسال می‌کند. می‌توانید فرض کنید که certificate او را از طریق `messenger.receive_certificate` دریافت کرده‌اید و او نیز certificate شما را در اختیار دارد. اگر پیش از این با او ارتباطی نداشته‌اید، جلسه را مطابق با مشخصات Signal با تولید کلیدهای لازم برای double ratchet راه‌اندازی کنید. در هر بار ارسال، sending chain (و در صورت لزوم root chain) را به‌روزرسانی کنید. یک header بسازید که شامل داده‌های لازم برای طرف مقابل جهت استخراج کلید جدید باشد. در نهایت، پیام را با استفاده از کلید ارسال جدید رمز کنید. **هر پیام باید با یک کلید ارسال جدید رمز شود.**

### ۲-۴. `messenger.receive_message(name, [header, ciphertext])`

این متد یک پیام رمزنگاری‌شده را از کاربری با نام مشخص‌شده دریافت می‌کند. می‌توانید فرض کنید که certificate او را از طریق `messenger.receive_certificate` دریافت کرده‌اید و او root key اولیه را با استفاده از کلید ElGamal موجود در certificate شما محاسبه کرده است.

اگر تاکنون با او ارتباطی نداشته‌اید، جلسه را با تولید کلیدهای لازم برای double ratchet مطابق با

مشخصات Signal راه‌اندازی کنید. در هر دریافت، receiving chain (و در صورت لزوم root chain) را با استفاده از اطلاعات موجود در header به‌روزرسانی کنید و پیام را با یک **کلید دریافت جدید** رمز گشایی نمایید. اگر نشانه‌ای از دستکاری یا خراب‌بودن پیام دریافت‌شده مشاهده شد، بلافاصله exception ایجاد کرده و اجرای برنامه را متوقف کنید. (یعنی مهاجم پیام رمزنگاری‌شده شما را دستکاری کرده است)

## ۳. سوالات تشریحی

لطفاً به سوالات زیر درباره پیاده‌سازی‌تان پاسخ دهید. (پاسخ‌های شما باید لازم و در عین حال کافی و شامل جزئیات مهم باشند.)

a) اگر آلیس و باب هیچ‌گاه کلید DH خود را به‌روزرسانی نکنند چه می‌شود؟ لطفاً پیامدهای امنیتی این تغییر را با توجه به Forward Secrecy و Break-in Recovery توضیح دهید.

b) مکالمه زیر را بین آلیس و باب در نظر بگیرید:

**A:** Hey Bob, can you send me the locker combo

**A:** I need to get my laptop

**B:** Sure, it's 1234

**A:** Great, thanks! I used it and deleted the previous message

**B:** Did it work

طولانی‌ترین زنجیره ارسال‌شده (sending chain) آلیس چه اندازه است؟ زنجیره باب چطور؟ توضیح دهید.

c) متأسفانه، در سناریوی بالا، دارث مدتی ارتباطات آلیس و باب را تحت نظر داشته و در نهایت موفق شده گوشی آلیس را هک کند و تمام کلیدهای او را درست قبل از ارسال پیام سوم بدزدد. با این حال، دارث نمی‌تواند ترکیب قفل (locker combination) را به دست بیاورد.

نام ویژگی امنیتی مرتبط با این اتفاق را بیان کنید، آن را توضیح دهید و بگویید که چگونه Double Ratchet این ویژگی را فراهم می‌کند.

d) کتابخانه PyCryptodome می‌تواند امضاهای دیجیتال را به روش‌های مختلفی از جمله ECDSA و RSA تولید کند.

برای هر دو روش ECDSA و RSA، موارد زیر را با هم مقایسه کنید:

- کدام کلیدها زمان بیشتری برای تولید نیاز دارند؟
- کدام روش زمان بیشتری برای تولید امضا نیاز دارد؟
- کدام امضا از نظر طول (length) بلندتر است؟
- کدام روش زمان بیشتری برای تایید امضا (verify) نیاز دارد؟

**نکته:** برای این سوال، می‌توانید اسکریپت `src/question_4_code.py` که در starter code در اختیار شما قرار داده شده است را اجرا کنید.

## - بخش دوم (IPsec)

در این بخش نیاز است روی ارتباط end-to-end بخش قبلی، IPSec را شبیه سازی کنید. نکات این امر در زیر توضیح داده شده است. IPsec مجموعه ای از پروتکل ها است که در لایه شبکه عمل می کند تا ارتباط امن بین دو نقطه پایانی را فراهم کند. محرمانه بودن، یکپارچگی و اصالت داده ها را بدون تغییر لایه برنامه تضمین می کند. به اجرای کلیدی IPsec در جدول زیر توجه کنید:

| توضیحات | المان |
|---|---|
| رمزگذاری و احراز هویت را فراهم می کند | **ESP** (Encapsulating Security Payload) |
| فقط احراز هویت را ارائه می دهد (بدون رمزگذاری) | **AH** (Authentication Header) |
| ایجاد و مدیریت انجمن های امنیتی (SAs) | **IKE** (Internet Key Exchange) |

## ۴. پیاده‌سازی IPsec بر روی ارتباط End-to-End

### ۴-۱. پیکربندی توصیه شده برای پیام رسانی ایمن

- **پروتکل**: ESP
- **حالت**: Transport
- **رمزگذاری**: AES-256-GCM
- **احراز هویت**: HMAC-SHA256
- **تبادل کلید**: IKEv2 با استفاده از کلیدها یا certificate های pre-shared
- **Replay Protection**: به طور پیش فرض در ESP از طریق اعداد ترتیبی فعال می شود.

### ۴-۲. IPsec شبیه سازی شده در پایتون (اجرای پروژه)

در این پروژه، IPsec با استفاده از رمزگذاری در سطح پایتون با AES-GCM تقلید شده است. یک کلید session ثابت با استفاده از HMAC مشتق شده و برای محافظت از پیام های UDP استفاده می شود. این امر محرمانه بودن و اصالت را بدون نیاز به پیکربندی در سطح سیستم عامل فراهم می کند.

## ۳-۴. پیاده سازی Transport API

`send_via_simulated_ipsec(dest_ip, dest_port, data)`

- Encrypts data using AES-GCM
- Sends encrypted packet over UDP

`receive_via_simulated_ipsec(bind_ip, bind_port)`

- Listens on given IP and port
- Decrypts incoming packet using shared key

## بخش سوم (پیاده سازی IPsec با استفاده از strongSwan) (امتیازی)

بعد از انجام بخش اول، مراحل زیر را طی کنید و گزارش دهید. شما نیازی به پیاده‌سازی مستقیم IPsec در پایتون ندارید. در عوض، باید از ابزارهایی مانند `ip xfrm` (بخشی از strongSwan) یا `setkey` بومی لینوکس استفاده کنید. برای این کار، پیشنهاد می‌شود پروژه خود را به این شکل تغییر دهید:

- رابط‌های مجازی امن (مانند eth0) راه‌اندازی کنید.
- ترافیک پیام را از طریق این رابط‌ها بین جفت‌های 127.0.0.1:<port> مسیریابی کنید.
- فرآیند پیکربندی و راه‌اندازی به صورت خودکار و از طریق فراخوانی‌های subprocess در پایتون انجام شود.

### ۵. IPsec با استفاده از strongSwan

#### ۵-۱. نصب strongSwan

```
1  sudo apt install strongswan
```

#### ۵-۲. نمونه فایل های config

```
/etc/ipsec.conf



config setup
  charondebug="ike 2, knl 2, cfg 2"

conn secure-messaging
  left=127.0.0.1
  leftsubnet=127.0.0.1/32
  leftid=alice
  right=127.0.0.2
  rightsubnet=127.0.0.2/32
  rightid=bob
  auto=start
  keyexchange=ikev2
  authby=psk
  ike=aes256-sha256-modp2048!
  esp=aes256gcm16!
```

```
/etc/ipsec.secrets


alice : PSK "shared_secure_password"
bob   : PSK "shared_secure_password"
```

### ۳-۵. فعال‌سازی IPsec

```
sudo ipsec restart
sudo ipsec up secure-messaging
```

## ملاحظات تمرین

مهلت تحویل: ۲۷ خرداد ماه

- تمام کدی که باید بنویسید باید در پوشه `src` باشد. لطفاً فایل‌های lib.py و test_messenger.py را ویرایش نکنید.
- سیستم پیام‌رسانی شما از کتابخانه‌ی *PyCryptodome* برای پیاده‌سازی رمزنگاری پایه استفاده می‌کند. با این حال، شما **نباید مستقیماً از توابع PyCryptodome استفاده کنید** (و کد ابتدایی هم مستقیماً از آن استفاده نکرده است). یک فایل پشتیبان به نام `lib.py` فراهم شده که wrappers برای توابع موردنیاز از PyCryptodome در اختیار شما می‌گذارد.
- حتماً کامنت های موجود در فایل `lib.py` را بررسی کنید تا نوع داده‌ها، ورودی‌ها و خروجی‌های توابع مختلف را به خوبی درک کنید.
- می‌توانید به فایل `tests/test_messenger.py` نگاهی بیندازید تا تست‌هایی که اجرا می‌شوند را ببینید. نوشتن تست‌های بیشتر برای اطمینان از عملکرد درست پیاده‌سازی‌تان اختیاری است. تست‌ها با استفاده از چارچوب `unittest` پایتون نوشته شده‌اند. (تست ها فقط برای بخش اول هستند.)
- تست‌ها برای تسهیل روند توسعه و فهم بهتر ساختار اولیه طراحی شده‌اند. پاس شدن آن‌ها ملاک نمره‌دهی نیست؛ تسلط کامل بر پیاده‌سازی و درک صحیح الگوریتم‌ها برای کسب نمره الزامی است.
- اگر برنامه شما متوجه دستکاری در داده‌های در حال انتقال شود (مانند ciphertext، امضاها و ...)، باید exception ایجاد کرده و اجرا را متوقف کند.
- پروژه به صورت گروه‌های دونفره انجام شود.
- لطفا تمام کدهای پروژه و گزارش خود را در قالب یک فایل Zip با فرمت زیر در سامانه Elearn بارگذاری کنید:

**NetSecProject_StudentID1_StudentID2**

- در صورت استفاده از منابعی غیر از کتاب مرجع در انجام تمرین، لطفا و حتما نام منبع خود را ذکر کنید. در صورت مشاهده شباهت غیرمعمول میان پاسخ‌های دو نفر یا در

صورتی که پاسخ‌ها برابر با محتوای منابعی غیر از کتاب مرجع باشد و نام منابع مورد استفاده ذکر نشده باشد، نمره‌ای برای شما منظور نخواهد شد.

- می‌توانید سوالات خود را از طریق ایمیل‌ها یا آیدی تلگرام زیر مطرح کنید:

بخش اول:

ab.eslami2001@gmail.com

بخش دوم و سوم:

aliabedini2001@gmail.com یا @abediniAli1

موفق باشید!