هو الحق

دانشکده‌ی مهندسی کامپیوتر

CE-40443 پاییز ۱۳۹۶

شبکه‌های کامپیوتری ـ تمرین سوم

استاد: مهدی جعفری

۱ مقدمه

هدف از این تمرین آشنایی با مفاهیم اولیه امنیت و privacy در شبکه و یک مثال از سیستم های پیشرفته‌تر و پیچیده‌تر ساخته شده براساس مفاهیم تدریس شده در درس شبکه است. به این منظور، شبکه‌ای مشابه شبکه Tor را تعریف و پیاده‌سازی خواهیم کرد. در این مسیر، از روش‌های متداول رمزنگاری، مسیریابی، پروتکل‌های شبکه و مفهوم *Onion Routing* [۱] استفاده می‌نماییم. برای حل این تمرین آشنایی اولیه با Tor ضروری نیست، اما شرکت در کلاس حل تمرین برای انجام ساده‌تر تمرین به شدت توصیه می‌شود.

۲ پیش‌زمینه

در این قسمت به بررسی نکات ضروری رمزنگاری و Onion Routing برای حل تمرین می‌پردازیم. مطالعه این قسمت خصوصاً در صورت عدم شرکت در کلاس حل تمرین ضروری است.

۱.۲ رمزنگاری

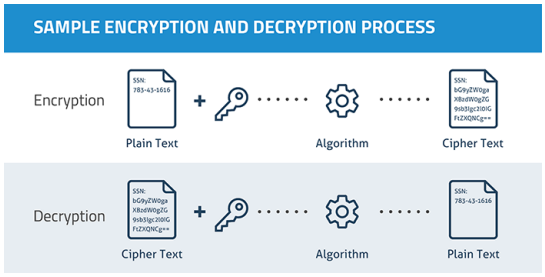
به طور کلی در رمزنگاری [۲] یک رشته بایت با یک الگوریتم ریاضی به رشته دیگری نگاشته می‌شود که به سادگی قابل بازگرداندن به رشته اولیه نیست. برای برگرداندن رشته جدید به پیام اولیه، دریافت کننده باید اطلاعات خاصی داشته باشد که معمولاً رشته سومی است که از آن به «کلید» یاد می‌شود.
به طور کلی می‌توان روش‌های رمزنگاری را به دو دسته متقارن و نامتقارن تقسیم نمود.

۱.۱.۲ رمزنگاری متقارن

در این روش‌ها کلید مورداستفاده هنگام رمزگذاری [۴] و رمزگشایی [۵] یکسان است. به این ترتیب هر کسی که بتواند رمز کردن پیغام را انجام دهد می‌تواند آن را رمزگشایی نیز بکند. از انواع این روش ها می‌توان به AES ، DES و حتی روش‌های ساده‌ای همانند Caesar Cipher اشاره کرد.

---

[۱] ترجمه تحت‌اللفظی *مسیریابی پیازی* نامناسب به نظر می‌رسد!
[۲] Cryptography
[۳] عکس برگرفته از http://www.skyhighnetworks.com
[۴] Encryption
[۵] Decryption



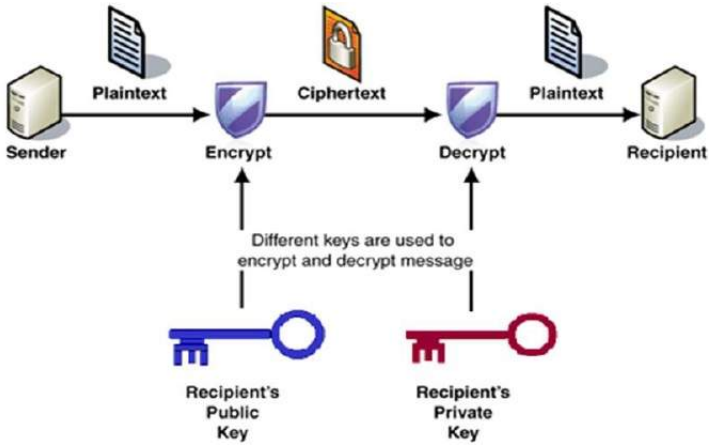
شکل ۱: تعریف سطح بالای رمزنگاری[۳]

اگر چه ممکن است این روش‌ها برای استفاده در شبکه کافی به نظر برسند، به سرعت می‌توان دریافت که کمبودهای زیادی در آنها وجود دارد. برای مثال اگر سایت‌های اینترنتی بخواهند **فقط** از روش‌های متقارن استفاده کنند، باید با هر کاربر خود کلیدی جداگانه برای رمزگذاری داشته باشند، در غیراین‌صورت کاربران می‌توانند مخابره یکدیگر را شنود نمایند. [۶]

۲.۱.۲ روش‌های نامتقارن

در این روش‌ها کلید به‌کار گرفته شده در رمزگذاری و رمزگشایی متفاوت هستند. یکی از مشهورترین روش‌های رمزنگاری نامتقارن RSA است. کلید رمزگذاری و رمزگشایی در RSA با یکدیگر رابطه ریاضی ویژه‌ای دارند که باعث می‌شود رشته‌های رمز شده با یکی، تنها به کمک دیگری قابل رمزگشایی باشند. به عبارت دیگر باداشتن کلید رمزگذاری به تنهایی، نمی‌توان عمل رمزگشایی را انجام داد.

اهمیت این موضوع آنجاست که کلید رمزگذاری می‌تواند به طور عمومی پخش شود، بدون آنکه امنیت روش رمزنگاری پایین بیاید. برای مثال، یک وبسایت معتبر می‌تواند کلید رمزگذاری را به طور عمومی برای تمام کاربران خود منتشر کند و تمام کاربران پیام های خود را با همان کلید رمزگذاری کنند. از آنجا که کلید رمزگشایی منتشر نشده و فقط سرور وبسایت آن را دارد، هیچ یک از کاربران در این روش نمی‌توانند پیام‌های یکدیگر را شنود کنند.



شکل ۲: رمزنگاری غیرمتقارن[۷]

با توجه به مثال بالا، به کلید رمزگذاری در این روش‌ها کلید عمومی یا Public Key و به کلید رمزگشایی کلید خصوصی یا Private Key گفته می‌شود. بدیهی است در صورت لو رفتن کلید خصوصی، رمزنگاری با این روش بی‌اثر خواهد بود.

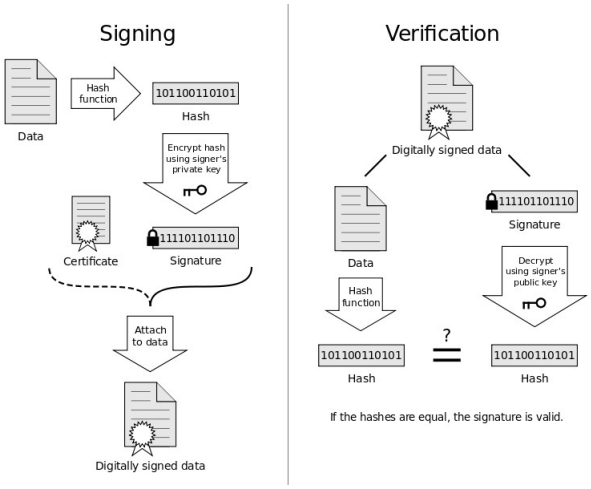
[۶] این بدان معنا نیست که در امنیت شبکه و سیستم به ندرت از روش‌های متقارن استفاده می‌شود. این روش‌ها به طور بسیار گسترده در رمزنگاری سیستم های کامپیوتری کاربرد دارند اما در بسیاری از موارد همانند مثالی که زده‌شد در کنار روش‌های نامتقارن استفاده می‌شوند.

[۷] عکس از Tutorialspoint

نکته: اگر چه در این تمرین برای سادگی به این موضوع توجهی نمی‌کنیم، در واقعیت به ندرت کل ارتباط بین دو نود شبکه یا یک فایل بزرگ با RSA رمزگذاری می‌شود. به طور معمول از RSA یا روش نامتقارن دیگر مورداستفاده تنها برای توافق روی یک کلید یک‌بارمصرف استفاده می‌شود که باقی مخابره با آن کلید و یک روش متقارن رمزگذاری می‌شود. دلیل این تکنیک، کند بودن شدید RSA با توجه به محاسبات سنگین ریاضی آن و وجود حمله های موثر شکستن رمز هنگام استفاده از RSA روی بلاک‌های بزرگ داده می‌باشد.

۲.۲ امضا

روش‌های امضا اهمیت بسیار زیادی در رمزنگاری سیستم‌های کامپیوتری دارند. به نوعی، امضاء عمل برعکس رمزگذاری غیرمتقارن است.
منظور از امضا در سیستم‌های کامپیوتری این است که بتوان اثبات کرد یک رشته بیت توسط فرد یا موجودیت مشخصی منتشر شده‌است. به این منظور، فرد امضا کننده ابتدا hash داده‌های موردنظر را گرفته و سپس آنرا رمزگذاری می‌کند، با این تفاوت که در این روش رمزگذاری با کلید خصوصی انجام شده و رمزگشایی با داشتن کلید عمومی قابل انجام است.
ایده امضا در این است که امضای یادشده را فقط کسی که کلید خصوصی را داشته باشد می‌تواند تولید کند اما همه مخاطبین (که کلید عمومی را در اختیار دارند) می‌توانند درستی آن را تایید کنند.



شکل ۳: امضا و تایید RSA[۸]

همانند رمزگذاری در قسمت قبل، کلید عمومی هرکس می‌تواند در اختیار تمام مخاطبین وی باشد. مخاطبین با دریافت پیام و امضای آن (که همان hash رمزگذاری شده است) می‌توانند hash داده‌ها را محاسبه کرده، به کمک کلید عمومی امضای آن را رمزگشایی نمایند و با مقایسه آن با hash محاسبه شده دریابند که پیغام موردنظر توسط فرد یادشده تایید شده‌است.

۳.۲ امنیت و حریم خصوصی در اینترنت

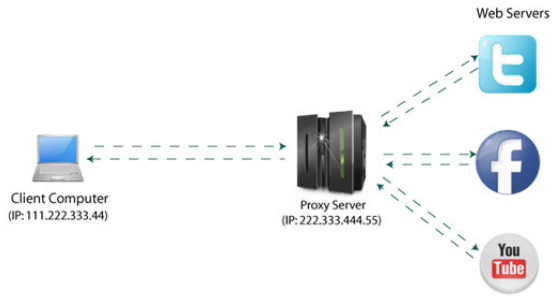
اینترنت به عنوان پروتکل مورد استفاده در شبکه‌های کامپیوتری به طور وسیع، با دید امنیت و حریم خصوصی طراحی نشده است. ساختار اینترنت به گونه‌ای است که به‌طور ذاتی خطرهای متعددی برای امنیت داده دربردارد. برای مثال، هنگام استفاده شما از اینترنت، موجودیت‌های بسیاری از جمله ISPها، دولت‌ها، افراد متصل به شبکه محلی شما و ... می‌توانند عملکرد شما را track کنند.
اگرچه وجود HTTPS و تمهیدات مشابه آن سطحی از حریم خصوصی را ایجاد می‌کنند، همچنان هویت شما و سرویس‌های مورد استفاده شما هنگام استفاده از اینترنت می‌تواند در اختیار دیگران قرار بگیرد. همچنین، سرویس‌های فعال در اینترنت

[۸] https://crypto.stackexchange.com/questions/12768/why-hash-the-message-before-signing-it-with-rsa

نیز می‌توانند توسط موجودیت‌های بیرونی مورد کنترل و تعقیب قرار بگیرند. به عنوان یک مثال، برخی سایت‌های خبری همانند Wikileaks در گذشته قربانی کنترل و سانسور غیرقانونی برخی دولت‌ها قرار گرفته‌اند.

۱.۳.۲ استفاده از Proxy

یکی از راه‌های تقویت حریم خصوصی، استفاده از proxy هاست. با گذراندن ترافیک اینترنت از یک گره دیگر در شبکه، می‌توان تعقیب آن را دشوارتر کرد.

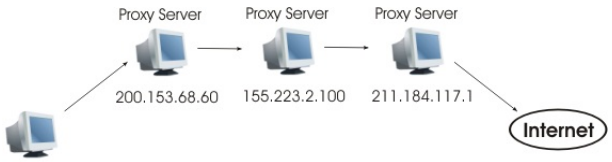


شکل ۴: [۹]استفاده از پراکسی در دسترسی به شبکه؛ افرادی که به دنبال track کردن عملیات کامپیوتر کارخواه باشند اطلاعات زیادی از گوش کردن به ترافیک آن به دست نمی‌آورند زیرا آدرس سایت‌های مورد بازدید به صورت رمز شده بین کارخواه و سرور پراکسی جابجا می‌شود و از دید ناظر بیرونی تمام ترافیک کارخواه به سمت سرور پراکسی می‌رود.

اگرچه این روش در نگاه اول موثر به نظر می‌رسد اما مسائل زیادی در رابطه با آن وجود دارد. برای مثال، خود سرور پراکسی تمام اطلاعات ترافیک کارخواه را می‌داند و کافیست حمله‌کننده به نحوی به آن دسترسی پیدا کند.
حمله‌های شبکه نیز به چنین سرورهایی نسبتاً ساده‌است و با تحلیل های آماری ساده می‌توان ترافیک کارخواه را با هزینه کم حدس زد.

۲.۳.۲ زنجیره پراکسی‌ها

با استفاده از چند پراکسی به صورت زنجیری می‌توان روش بالا را به لحاظ امن و خصوصی بودن تقویت کرد. برای مثال با گذراندن ترافیک از چند سرور موجود در کشورهای مختلف، می‌توان تعقیب ترافیک توسط ISP ها و دیگران را به شدت دشوارتر نمود.



شکل ۵: [۱۰]استفاده از زنجیره‌ای از پراکسی‌ها

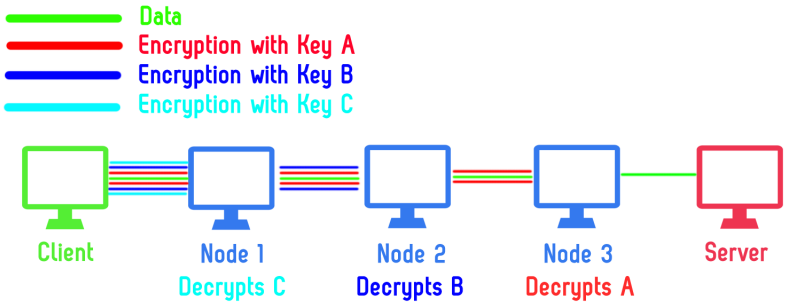
ایراد این روش آنجاست که سرورهای میانی همچنان اطلاعات زیادی در رابطه با ترافیک کاربر دارند. برای مثال گره میانی اول در تصویر بالا اطلاعات کل ترافیک کاربر را در اختیار دارد.

۳.۳.۲ Onion Routing

به طور کلی هدف Onion Routing تقویت مدل زنجیری قسمت قبل است. در این روش، می‌خواهیم کاری کنیم تا گره‌های میانی حداقل اطلاعات ممکن را در رابطه با ترافیک عبوری داشته باشند. ایده کلی در Onion Routing این است که با معرفی

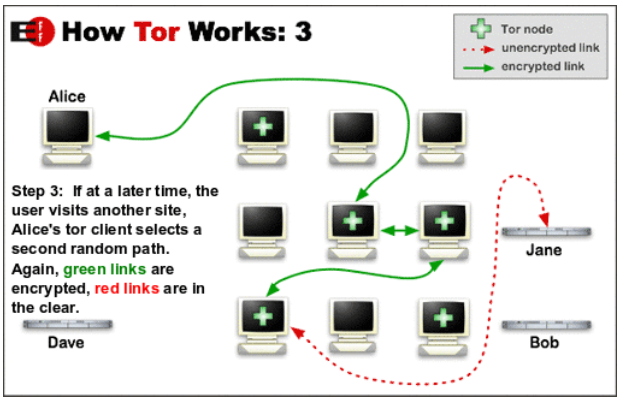
[۹] عکس برگرفته از وبسایت Hotspotshield
[۱۰] عکس از https://7inverse.wordpress.com/tag/chaining-proxies/

کردن لایه‌های متعدد رمزنگاری، کاری کنیم تا گره‌های شبکه تنها آدرس گره بعدی خود را بدانند. به این ترتیب اگر تعداد گره‌های میانی کافی باشد، هیچ یک از پراکسی‌ها نمی‌تواند هم‌زمان مبدأ و مقصد بسته در حال عبور را مشخص کند.
هر گره در شبکه Tor یک جفت کلید عمومی و خصوصی دارد که کلید عمومی آن برای گره‌های دیگر مشخص است. به این‌ترتیب می‌توان بسته‌ها را طوری رمزنگاری کرد که فقط یک گره مشخص بتواند آن را بخواند. فرستنده بسته، قبل از ارسال آن از طریق گره‌های میانی، آن را به ترتیب برعکس گره‌ها برای هرگره رمزنگاری می‌کند. هرگره، تنها مجاز است آدرس گره بعدی را بداند تا هیچ گرهی از کل مسیر باخبر نباشد.



شکل ۶: [۱۱]بسته ارسال شده در شبکه Tor چندین بار روی هم رمزنگاری می‌شود. هر گره میانی بسته دریافتی را با کلید خصوصی خود رمزگشایی می‌کند اما مجددا با بسته‌ای رمزشده برای گره بعدی (به همراه آدرس آن) روبرو می‌شود. نام *مسیریابی پیازی* در همین ویژگی ریشه دارد که بسته همانند پیاز لایه‌های مختلفی دارد که در هر گره یک لایه از آن باز می‌شود. تنها بسته‌ای که پکت IP نهایی را می‌بیند گره میانی آخر است که همچنان از آدرس فرستنده بی‌خبر است.

با این سازوکار، هر گره تنها می‌تواند بسته در حال انتقال را به فرمت قابل استفاده برای گره بعدی در آورد و برای آن ارسال کند. به جز گره میانی آخر، گره‌های میانی تنها آدرس گره بعدی را از بسته در می‌یابند. به همین دلیل حریم خصوصی کارخواه فرستنده حفظ می‌شود، چرا که هیچ‌کس در شبکه خبرندارد چه کسی برای چه کسی بسته IP را ارسال کرده‌است.

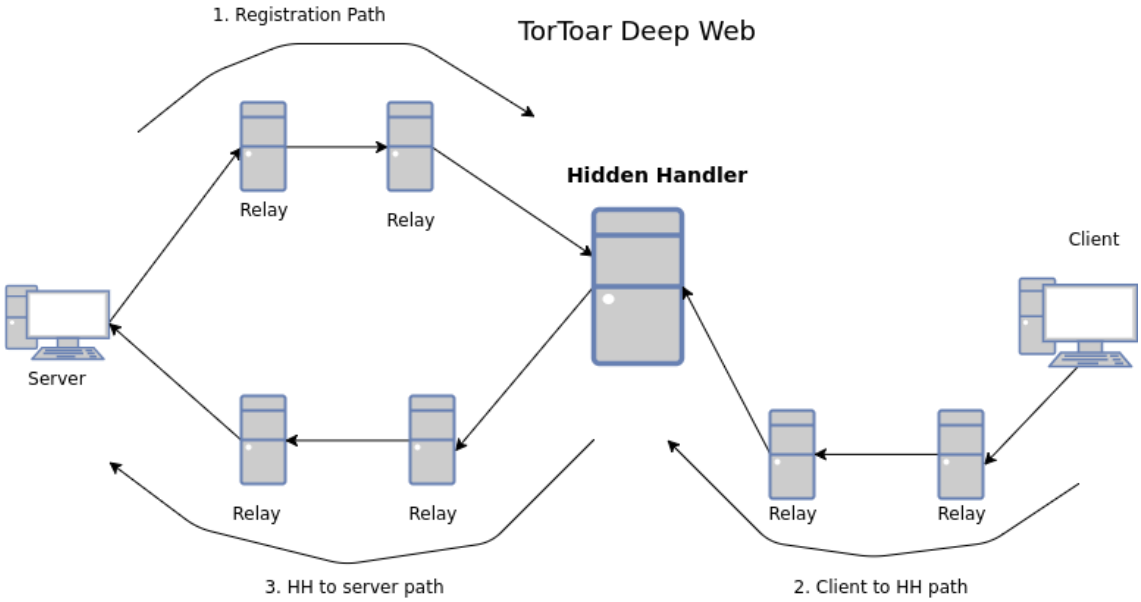


شکل ۷: [۱۲]شبکه Tor؛ تنها گرهی که پکت فرستنده را به‌طور کامل می‌بیند پراکسی آخر است که از هویت و آدرس فرستنده آن بی‌خبر است.

[۱۱] عکس از https://teamultimate.in/how-tor-works-complete-guide/
[۱۲] عکس از https://www.torproject.org/about/overview.html.en

۴.۳.۲ Deep Web

اگرچه سازوکار قسمت قبل می‌تواند حریم خصوصی فرستنده را تا حد زیادی محافظت کند، همچنان کارگزار موردنظر فرستنده محافظت‌نشده است. به عبارتی، ترافیک آن می‌تواند کنترل شود و به طریق‌های متفاوت از فعالیت آن جلوگیری شود.
راه حل شبکه Tor برای این مسئله روشی است که به طور عمومی با نام Deep Web شناخته می‌شود. به طور خلاصه، در این روش تمهیداتی صورت می‌گیرد که کارگزار مقصد به طور مستقیم از اینترنت قابل دسترسی نباشد و از طریق یک گره میانی خاص ارتباط با آن از طریق شبکه Tor امکان‌پذیر باشد.



شکل ۸: برای عملیاتی کردن Deep Web ابتدا سرویس‌دهنده خود را در یک گره میانی ویژه که Hidden Handler می‌نامیم رجیستر می‌کند و یک مسیر Tor برای فرستادن پیام به خود را معرفی می‌کند. سپس، کارخواه از طریق یک مسیر Tor به Handler Hidden متصل شده و از آنجا با ارائه URL سرویس‌دهنده که درواقع یک public-key است، بسته خود را به دست کارگزار می‌رساند.

توجه کنید که HH (Hidden Handler) آدرس دقیق کارگزار را ندارد بلکه تنها آدرس گره بعدی در مسیر از خود به کارگزار را هنگام رجیستر شدن کارگزار دریافت می‌کند. آدرس باقی گره‌ها (و خود کارگزار) به صورت رمزگذاری شده به HH داده می‌شود. به این ترتیب HH می‌تواند برای کارگزار بسته ارسال کند بدون آنکه از آدرس واقعی آن خبر داشته باشد. این موضوع حریم خصوصی کارگزار را حفظ می‌کند و موجودیت‌های خارجی نمی‌توانند در عملکرد آن به آسانی وقفه ایجاد کنند چرا که حتی آدرس دقیق آن برای همگان مشخص نیست.

توجه: جزئیات روش‌های به کارگرفته شده در این تمرین ممکن است با آنچه در حقیقت در Tor انجام می‌گیرد بسیار متفاوت باشد. برای اطلاع بیشتر از عملکرد دقیق Tor به منابع بیرونی مراجعه کنید.

۳ آماده‌سازی

تمپلیت انجام این تمرین به زبان پایتون است. شما ابتدا بایستی داده شده را در مخزن خود قرار دهید:

```
cp -R <HANDOUT DIRECTORY>/hw3 <YOUR REPO DIRECTORY>/
cd hw3
```

سپس پکیج‌های مورد نیاز تمرین را با pip [۱۳] نصب کنید. [۱۴] دقت کنید نسخه پایتون پروژه بایستی ۳ به بالا باشد. اگر پایتون پیش‌فرض سیستم‌عامل شما نسخه ۲ است به جای pip و python از دستورات pip۳ و python۳ استفاده کنید یا یک virtualenv پایتون ۳ ایجاد نمایید.

```
sudo pip install -r requirements.txt # omit "sudo" if using virtualenv
```

پس از نصب نیازمندی‌های تمرین، آماده شروع به کار هستید.

۴ وظیفهٔ شما و پیاده‌سازی

برای سادگی بیشتر، در این تمرین از وظایف شبکه‌ای ساختن سوکت و فرستادن پیام‌های TCP/IP صرف‌نظر شده و از یک شبکه شبیه‌سازی شده به ابتدایی‌ترین شکل ممکن استفاده می‌شود. قالب پروژه شامل کلاس‌ها و توابعی است که برخی از آنها پرشده و باقی توسط شما پر می‌شود. برای نمره‌دهی، توابع هم به صورت جدا و هم به صورت جمعی تست می‌شوند و از این رو در صورتی که یکی از توابع عملکرد کل سیستم را مختل کند تمام تست‌های شما fail نمی‌شود و در حق کسی اجحاف نخواهد شد.
توابعی که بایستی توسط شما پر شوند در کد با پیام کامنت شده زیر مشخص شده‌اند. داخل کد داکیومنتیشن نسبتاً مفصلی برای هر تابع و کلاس نیز موجود است و عملکرد هر تابع و ماهیت و type ورودی و خروجی‌های اکثر توابع نیز مشخص شده‌است. با این‌حال در صورت شک داشتن در مورد عملکرد یا ماهیت هریک از توابع یا متغیرها می‌توانید سوال خود را از طریق piazza یا ایمیل دستیاران آموزشی مطرح نمایید.

```
# TODO this is filled by the student
```

در قالب تمرین در فایل‌های `utils.py` و `crypto.py` توابع کمکی برای شما نیز تدارک دیده‌شده است تا رمز کردن و برخی عملیات دیگر را ساده‌تر انجام دهید. در اینجا برخی جزئیات برنامه‌نویسی تمرین را مرور می‌کنیم و اختصاراً به بررسی آنچه توسط شما بایستی نوشته شود می‌پردازیم.

۱.۴ bytes

در پایتون نوع‌داده‌ای به نام `bytes` وجود دارد که API آن بسیار مشابه `string` پایتون است اما رشته‌های بایتی را نمایش می‌دهد و به همین دلیل در پروژه‌های شبکه، سوکت و پایپ معمولاً از آن استفاده می‌شود. یک رشته بایتی می‌تواند به سادگی به صورت زیر تعریف شود:

```
bytes_string = b"I love this assignment very much. Seriously."
```

دقت کنید دستورهای مشابه دستورات `string` همانند `split` و `replace` برای `bytes` نیز وجود دارد اما ورودی آنها نیز به جای `str` باید `bytes` باشد. می‌توانید قبل از شروع، با این توابع در shell پایتون کمی بازی کنید:

```
>>> bytes_string = b"I love this assignment very much. Seriously."

>>> bytes_string[2:22]
b'love this assignment'

>>> bytes_string.replace(b"e", b"3")
b'I lov3 this assignm3nt v3ry much. S3riously.'
```

[۱۳] pip یک package-manager رسمی برای پایتون است که به کمک آن می‌توان پکیج‌ها و کتابخانه‌های موردنیاز پروژه‌ها را به آسانی نصب نمود.
[۱۴] در صورت آشنایی با virtualenv پیشنهاد می‌کنیم یک virtualenv جدید با نسخه پایتون ۳ ایجاد کنید و پس از فعال سازی آن دستور نصب را بدون sudo اجرا کنید. برای آشنایی بیشتر با virtualenv می‌توانید این لینک را مطالعه کنید.

۲.۴ رمزنگاری

برای رمزنگاری ساده‌تر از پکیج `rsa` پایتون استفاده می‌کنیم که در requirement های پروژه آن را نصب کرده‌اید. در این پکیج دو کلاس مهم `PublicKey` و `PrivateKey` وجود دارند که بدیهتاً کلیدهای عمومی و خصوصی RSA را مدل می‌کنند. برای آشنایی بیشتر با این پکیج پیشنهاد می‌شود صفحه راهنمای نسبتاً کوتاه آن را در این صفحه مطالعه کنید. مثال زیر یک رمزگذاری و رمزگشایی ساده با این پکیج را نمایش می‌دهد:

```
import rsa
pubkey, privkey = rsa.newkeys(2048)
message = b"TorToar"
ciphertext = rsa.encrypt(message, pubkey)
plaintext = rsa.decrypt(crypto, privkey)
assert message == plaintext
```

برای رمزگذاری و رمزگشایی رشته‌های بیتی بزرگ‌تر از ۲۵۴ بایت، از توابع `blob_rsa_dec(mbytes, privkey)` و `blob_rsa_enc(mbytes, pubkey)` تدارک دیده‌شده در ماجول `crypto.py` استفاده کنید.
همچنین در ماجول `utils` دو دستور `key_to_bytes(pubkey)` و `bytes_to_key(mbytes)` برای تبدیل آبجکت‌های `rsa.PublicKey` به رشته بیتی قابل استفاده در پکت و برعکس تعریف شده‌اند.

۳.۴ پروتکل TorToar

پروتکل این تمرین، نسخه‌ای بسیار ساده‌شده از شبکه Tor به نام *TorToar* را پیاده‌سازی می‌کند و بیانگر مفاهیم توضیح داده شده در قسمت پیش‌زمینه است.

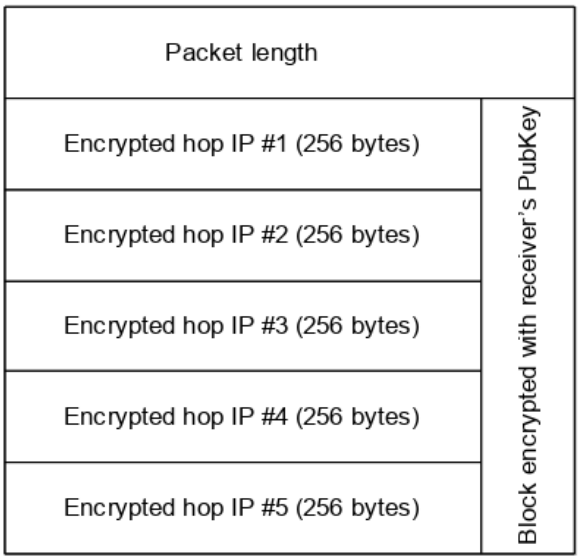
۱.۳.۴ ساختار پکت

پکت‌های TorToar شامل دو قسمت سرآیند (Header) و بدنه (Body) می‌شوند. سرآیند تمام پکت‌های پروتکل دارای قالبی یکسان هستند.

۲.۳.۴ ساختار سرآیند

سرآیند پکت‌های TorToar شامل طول packet و آدرس گره‌های میانی که پکت باید ازآنها بگذر می‌شود. داخل سرآیند، پنج فیلد برای مشخص کردن گره‌های میانی یا همان hop ها وجود دارد. توجه کنید ما دوست داریم حداقل اطلاعات ممکن را به گره‌های میانی بدهیم، در نتیجه ترجیح می‌دهیم حتی مشخص نباشد چند hop دیگر تا مقصد باقی مانده‌است. به همین دلیل، با هرتعداد hop موجود تا گره بعد، ابتدا از بالا فیلدهای hop ها پر می‌شوند و اگر فیلدی خالی باقی بماند با بیت‌های رندوم پر می‌شود. [۱۵]
هر فیلد، شامل آدرس IP یک گره‌میانی است که با کلید عمومی گره‌میانی قبل از آن در مسیر رمزگذاری شده‌است. هر گره میانی، پس از دریافت و parse کردن سرآیند پکت، ابتدا اولین فیلد را با کلید خصوصی خود رمزگشایی می‌کند تا آدرس hop بعدی را پیدا کند. سپس فیلد ها را یکی به بالا شیفت داده و فیلد آخر را با بیت‌های رندوم پرمی‌کند و پکت را برای گره بعدی ارسال می‌کند.
توجه کنید، قبل از ارسال پکت کل بلوک hop ها (هر ۵ فیلد) نیز با کلیدعمومی گیرنده پکت رمزنگاری می‌شود و مشابهاً هنگام دریافت نیز ابتدا باید بلوک یادشده با کلید خصوصی گیرنده رمزگشایی شود. [۱۶]
نکته مهم: دقت کنید اندازه های ذکرشده در شکل بالا مربوط به ساختار پکت قبل از رمزگذاری قسمت hop های سرآیند هستند. همانطور که گفته شد در هربار اجرای `rsa.encrypt` حداکثر می‌توان ۲۵۴ بایت را رمزگذاری کرد. پس از رمزگذاری قسمت hop های سرآیند، اندازه آن حدود ٪۸ افزایش می‌یابد.
اندازه کل سرآیند برابر `4 + 256 * ceil(5 * 256 / 254) = 1540 byte` خواهد بود. (چرا؟)

[۱۵] می‌توانید از `os.urandom(bytes_count)` استفاده کنید
[۱۶] اگر این کار انجام نشود با تحلیل هوشمندانه داده‌های شبکه می‌توان شیفت خوردن فیلدها به بالا را مشاهده کرد و مسیر پکت را ردیابی نمود.



شکل ۹: ساختار سرآیند پکت TorToar

هنگام ساختن پکت آغازین توسط کارخواه، مسیر از قبل توسط خود کارخواه تعیین شده است. کارخواه حداکثر ۴ hop میانی قبل از گیرنده مشخص می‌کند. فرض کنید فرستنده را S، این hop ها را A۱ تا A۴ و گیرنده را D نامگذاری کنیم. فیلدهای یادشده به این صورت پر می‌شوند:

```
rsa(A2.ip, A1.pubkey)
rsa(A3.ip, A2.pubkey)
rsa(A4.ip, A3.pubkey)
rsa( D.ip, A4.pubkey)
rsa(  EOH,  D.pubkey)
```

که منظور از eoh آدرس مشخصی است که گره‌ها با دیدن آن در فیلد اول به عنوان آدرس hop بعدی، متوجه می‌شوند hop دیگری وجود ندارد و پکت برای خودشان ارسال شده. در پروتکل TorToar این آدرس برابر `0.0.0.0` است. [۱۷]

۳.۳.۴ ساختار بدنه

در TorToar دو نوع پکت وجود دارد:

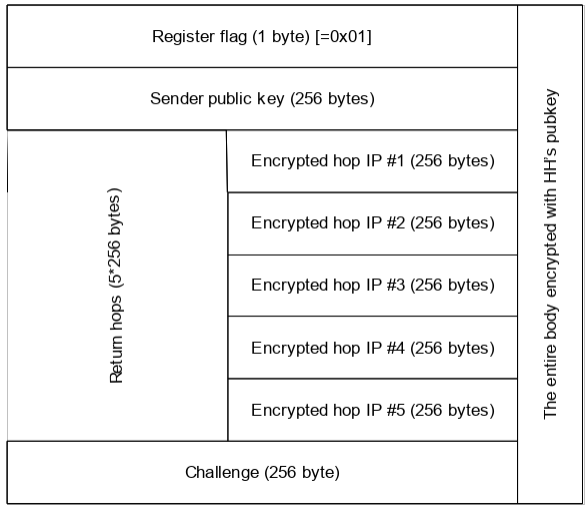
۱. پکت Registration

۲. پکت Data

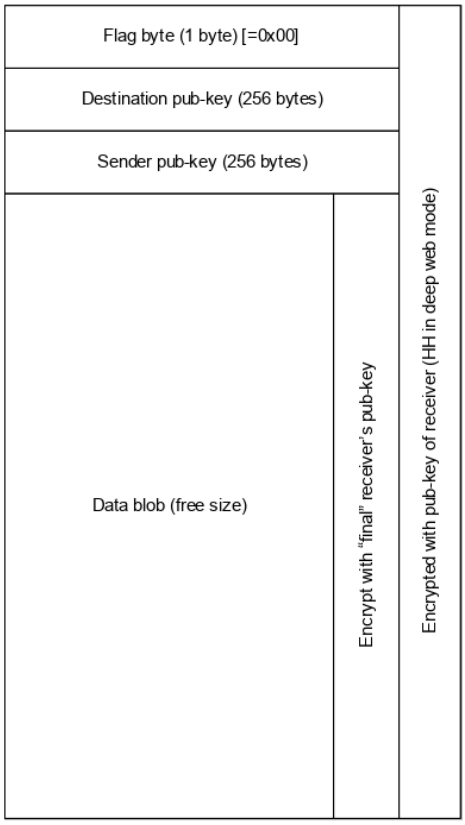
همانطور که گفته شد برای استفاده از قابلیت Deep Web نیاز است سرویس‌دهنده خودش را در یکی از گره‌های شبکه که آن را Hidden Handler نامیدیم رجیستر کند. [۱۸]

[۱۷] می‌توانید برای ساختن آن `core.Relay.eoh(dest_pk)` را استفاده کنید یا کد آن را کپی نمایید.

[۱۸] در این تمرین برای سادگی، گره‌ها همگی کد یکسانی دارند. به این معنا که همه گره‌های شبکه می‌توانند به عنوان کارگزار، کارخواه، گره میانی عادی یا گره میانی Hidden Handler فعالیت کنند.



(آ) ساختار بدنه پکت رجیستر



(ب) ساختار بدنه پکت دیتا

شکل ۱۰: ساختار بدنه در پروتکل TorToar

بایت اول بدنه هر پکت نوع آن را مشخص می‌کند. اگر مقدار این بایت `0x01` باشد آن پکت، پکت رجیستر و در غیر این صورت اگر مقدار آن `0x00` باشد پکت دیتا خواهد بود. [۱۹] ساختار بدنه بر اساس نوع پکت یکی از دو حالت نمایش داده شده در شکل ۱۰ را دارد.

نکته بسیار مهم: پکت دیتا ممکن است مستقیماً برای یک گره ارسال شود یا به جای آن برای یک Hidden Handler ارسال شود و HH آن را مجدداً برای گیرنده نهایی ارسال کند. گیرنده بر اساس فیلد `Destination pub-key` تشخیص می‌دهد که آیا مخاطب بسته خودش است یا باید در نقش HH ظاهر شود و آن را برای یکی از گره‌های رجیستر شده بفرستد. (می‌توانید فرض کنید پکتی برای گرهی رجیستر نشده ارسال نمی‌شود و لازم نیست این خطا را هندل کنید) نکته مهم اینجاست که کل بدنه همواره برای گیرنده رمزگذاری می‌شود در حالی که قسمت *Data blob* داخل آن برای **گیرنده نهایی** رمزگذاری می‌شود که در حالت عادی همان گیرنده و در حالت Deep Web کارگزار رجیستر شده است. به طور خلاصه Data Blob با همان کلیدعمومی `Destination pub-key` رمزگذاری می‌شود و در بدنه قرار می‌گیرد سپس کل بدنه با کلید عمومی گیرنده رمزگذاری می‌گردد. [۲۰]

challenge در رجیستر کردن

اگر کسی pub-key پنهان یا اصلی فرد دیگری را در یک HH رجیستر کند، این موضوع ناامنی خاصی ایجاد نمی‌کند چرا که

[۱۹] چرا این بایت را در سرآیند قرار ندادیم؟ چون نمی‌خواستیم گره‌های میانی از نوع پکت خبر داشته باشند. مسلماً این پروتکل تنها راه یا حتی بهترین راه ممکن برای پیاده‌سازی نیازمندی‌های ما نیست.

[۲۰] **گیرنده** و **گیرنده نهایی** در حالت ارسال عادی یک نفر هستند و در حالت Deep Web گیرنده HH است و گیرنده نهایی همان کارگزار پنهان است.

فرض براین است که وی به priv-key لازم برای فهمیدن پیام‌های آن را ندارد. با این حال اگر تایید نشود که فرد رجیسترکننده به راستی کلید عمومی خودش را رجیستر می‌کند، وی می‌تواند با منحرف کردن ترافیک نودهای دیگر به سمت خود در شبکه اختلال ایجاد کند. به این منظور هنگام رجیستر کردن از کارگزار خواسته می‌شود زمان فعلی شبکه را با کلید خصوصی خود امضا کند تا مشخص شود هویت کس دیگری را جعل نمی‌کند. [۲۱]

۴.۴ وظیفه شما

به‌طور کلی شما بایستی توابع مربوط به ۴ قسمت را پر کنید:

۱. ساختن مسیر

۲. تبدیل پکت به دنباله بایت‌ها براساس پروتکل و برعکس

۳. هدایت (routing) پکت

۴. ارسال درخواست های رجیستر و دیتا به کمک مورد ۲

گره‌های شبکه در کلاسی به نام `Relay` تعریف شده‌اند. همانطور که گفته‌شد برای سادگی گره‌های شبکه همگی مشابه یکدیگر هستند و می‌تواند در نقش کارگزار، کارخواه، گره میانی یا گره میانی HH فعالیت کنند.
عمده قسمت پروتکل در ماجول `packet.py` تعریف شده که در آن کلاس‌هایی برای پکت، سرآیند و بدنه وجود دارند. پیشنهاد می‌شود مستندسازی داخل کد را نیز مطالعه نمایید و از توضیحات ذیل برای درک بهتر ساختار کد استفاده کنید.

Relay ۱.۴.۴

آبجکت‌های کلاس `Relay` تنظیمات شبکه را هنگام ساخته شدن دریافت می‌کنند. کلاس `RelayConfig` که همین تنظیمات را مدل می‌کند، شامل **لیست تمام گره‌های مشهود برای گره موردنظر** و **گراف شبکه** می‌شود.

منظور از آدرس گره یا `RelayAddress` یک جفت IP Address و Public Key است. فهرست گره‌ها در `RelayConfig` در واقع لیستی از همین `RelayAddress` هاست.

همچنین گراف شبکه نیز در واقع لیستی از tuple ها به فرمت زیر است که هرکدام یک یال شبکه را مشخص می‌کنند:

```
(<start node IP address>, <end node IP address>, <network latency>)
```

بنابراین گراف شبکه وزن‌دار و جهت‌دار فرض می‌شود.

توجه داشته باشید هر گره یک **کلید عمومی اصلی** دارد که برای همه دانسته‌شده است و برای پکت فرستادن به آن استفاده می‌شود. در ادامه، می‌توان به صورت optional به `Relay` یک کلید عمومی پنهان نیز داد تا هنگام استفاده از Deep web کسی نتواند دریابد public key رجیستر شده در HH مربوط به کدام گره است. [۲۲]

packet.py ۲.۴.۴

هریک از کلاس‌های `Packet` ، `Header` ، `Body` و زیرکلاس‌های آن دارای یک متود `to_bytes()` و یک متود `from_bytes()` هستند که برای تبدیل رشته بایت‌های دریافت شده در شبکه به آبجکت پکت و برعکس استفاده می‌شوند. در `to_bytes()` کلاس `Packet` بدیهتاً از `to_bytes()` کلاس‌های دیگر استفاده می‌شود و همین موضوع برای `from_bytes()` نیز صادق است.

[۲۱] هنگام پیاده سازی می‌توانید از دستور `Relay.challenge()` استفاده کنید.

[۲۲] به یاد داشته باشید که در `RelayConfig` مپ آدرس IP و کلید عمومی اصلی به یکدیگر وجود دارد. از این رو، اگر HH کلید عمومی اصلی سرور رجیستر شده را بداند در عمل آدرس فیزیکی آن را نیز می‌داند. به همین دلیل گره‌هایی که سرویس Deep web دارند بایستی با کلیدعمومی پنهان خود آن سرویس را ارائه بدهند.

در مستندسازی هرکدام از این دستورها ورودی‌ها و خروجی هریک به همراه نوع‌داده آنها توضیح داده شده‌است. به نکات یاد شده در این مستندسازی‌ها حتماً توجه فرمایید.

نکات:

- در ورودی و خروجی `PacketBody.to_bytes()` و `PacketBody.from_bytes()` که توسط فرزندان آن پیاده‌سازی می‌شود، بایت flag که نوع بدنه را مشخص می‌کنند قرار ندارد و بایت‌های بدنه از بعد از آن بایت توسط این کلاس هندل می‌شود. خود بایت فلگ در دستورات کلاس `Packet` اضافه می‌شود. همچنین رمزگذاری و رمزگشایی نهایی بدنه در `Packet` هندل می‌شود و شما در `PacketBody` با رشته‌های بایتی عادی سروکار دارید. (Data Blob به صورت رمز شده به این کلاس داده می‌شود و خود کلاس هیچگونه انجام نمی‌دهد)

- کلاس `PacketBody` دارای سه فرزند است. مشخصاً عملکرد DataPacketBody و RegisterPacketBody مربوط به دو نوع بدنه پکت‌ها در TorToar است. اما فرزند سوم این کلاس RawPacketBody مربوط به حالتی است که بدنه پکت برای گره میانی نامعلوم است. از این رو هیچ ساختاری برای آن قائل نیستیم و صرفاً همان بایت‌هایی که دریافت کرده‌ایم را مجدداً به گره‌های دیگر دست‌به‌دست می‌کنیم.

- از آنجا که در پروتکل ما قسمت length پکت رمزگذاری نشده است و قسمت hops آن رمزگذاری می‌شود، برخلاف `PacketBody` باستی `Header` خودش رمزگشایی و رمزگذاری قسمت `hops` را هندل کند. از این رو توابع این کلاس کلید عمومی یا خصوصی مورد نیاز خود را نیز دریافت می‌کنند.

- `from_bytes()` در `Packet` میتواند یک یا دو کلید خصوصی به عنوان ورودی دریافت کند که اولی کلید خصوصی اصلی و دومی کلید خصوصی هویت پنهان `Relay` فراخواننده است. این بدان معناست که ابتدا باید امتحان شود که با کلید خصوصی هویت اصلی گره می‌توان پیغام را رمزگشایی کرد یا خیر و در غیر این صورت اگر کلید خصوصی پنهان نیز پاس داده شده بود، آن را تست می‌کنیم.

- در متود `to_bytes()` کلاس `Packet` کلیدعمومی گیرنده و کلید عمومی هاپ بعدی داده می‌شود که اولی برای رمزگذاری قسمت بدنه و دومی برای رمزگذاری قسمت سرآیند کاربرد دارد.

core.py ۳.۴.۴

در کلاس `Relay` توابع زیر بایستی پر شوند:

- `build_circuit()`

این تابع دو گره شبکه (مشخص شده با آدرس IP) را دریافت می‌کند و یک مسیر با خصوصیات زیر بین آن دو پیدا می‌کند:

  - حداقل ۲ نود میانی بین مبدا و مقصد باشد
  - نودهای میانی حداقل در دو کشور متفاوت باشند
  - تعداد هاپ‌های مسیر کمینه باشند (چون هنگام گذشتن از یک گره جدید معمولاً latency زیادی برای رمزگشایی و رمزگذاری مجدد اضافه می‌شود)
  - بین مسیرهای دارای ویژگی‌های فوق، طول وزن‌دار مسیر کمینه باشد (وزن های گراف همان زمان ping یا latency شبکه هستند)

برای دریافتن کشوری که هر آدرس IP در آن قرار دارد از دستور `self.get_ip_country()` استفاده کنید. دقت کنید تابع بایستی نود ابتدا و انتهایی را نیز در بر داشته باشد. همچنین به طور کلی تمام گره‌ها در این تابع با آدرس IP شان مشخص می‌شوند.

- `relay_packet()`

این دستور هنگامی فراخوانی می‌شود که مشخص شود مقصد پکت دریافت شده گره دیگری است. به این ترتیب بایستی سرآیند آن به‌روزرسانی شود و برای گره بعدی ارسال گردد.

نکته: برای ارسال پکت از دستور `self.netman.convey_packet()` تعریف شده در `net.py` استفاده نمایید.

- `receive_packet()`

این دستور هنگامی فراخوانی می‌شود که مشخص شود مقصد پکت همین گره بوده یا به عبارت دیگر آدرس هاپ بعدی `b"0.0.0.0"` باشد. دقت کنید ممکن است گره فعلی در نقش HH فعال باشد و همچنان نیاز باشد تا محتوای پکت برای گره دیگری ارسال گردد.

- `send_data_hidden()`

در این تابع بایستی به کمک توابع پیاده‌سازی شده در `packet.py` براساس ورودی‌ها پکت مناسبی برای ارسال به روش `Deep web` ایجاد کنید و آن را به مقصد مناسب ارسال نمایید.

- `send_data_simple()`

در این تابع بایستی مشابه قبلی می‌بایست پکتی از نوع دیتا اما به صورت عادی (غیر Deep web) با توجه به ورودی‌ها ایجاد و ارسال کنید.

- `register_on()`

مشابه دو تابع قبل در اینجا نیز از ماجول `packet.py` استفاده می‌شود. براساس ورودی‌ها و به کمک کلاس `Packet` یک پکت رجیستر برای مقصد ارسال نمایید. برای پر کردن فیلد `challenge` نیز همانطور که قبلاً در پاورقی گفته‌شد از تابع `self.challenge()` استفاده نمایید.

دستور `on_data()` مربوط به زمانی است که گره فعلی دریافت کننده **نهایی** یک پکت باشد. به عبارت دیگر هنگامی که گره در مقام HH پکتی را دریافت می‌کند که باید برای کس دیگری ارسال شود، این تابع صدا نمی‌شود. محتوای این تابع هنگام جاج کردن تغییر می‌کند از این رو می‌توانید آن را به دلخواه عوض کنید.

۵ امتیازدهی

امتیازدهی به صورت خودکار انجام می‌شود و جاج تمرین یک هفته پس از آپلود شدن مستند آن شروع به کار خواهد کرد. اما برای تست کردن کد خود می‌توانید از توابع تعریف شده در فایل `run.py` استفاده کنید.

۶ نکات

- در این تمرین و سایر تمرین‌های درس، با هرگونه تقلب شدیداً برخورد خواهدشد. کدهای ثبت شده به صورت خودکار برای تقلب بررسی می‌شوند.
- درصورت هرگونه ابهام در رابطه با عملکرد کلاس‌ها و متودها، پروتکل و یا مفاهیم تمرین سوال خود را با دستیاران درس مطرح نمایید و از حدس‌های ریسک‌دار خودداری کنید.
- به ازای هر روز ارسال زودتر از موعد تمرین ٪۵ نمره امتیازی به شما تعلق خواهد گرفت. (حداکثر تا٪۳۵)
- ددلاین ارسال تمرین ساعت **۲۳:۵۹** روز **جمعه ۱۳ بهمن ۱۳۹۶** می‌باشد. پس از آن به ازای هرروز، ٪۲۰ از نمره شما کاسته خواهد شد. توجه داشته باشید که به خاطر فرصت محدود برای ارسال نمرات به هیچ عنوان این تاریخ قابل تغییر نیست.
- در صورت داشتن هرگونه سوال و مشکل، می‌توانید از طریق piazza و یا ایمیل‌های `imakbari@gmail.com` و `hooman.shaah@gmail.com` آنها را مطرح کنید.

- از ارسال پاسخ در پیاتزا و گروه‌های تلگرام و سایر منابع عمومی خودداری کنید.

موفق باشید