به نام خداوند بخشنده مهربان

دانشگاه صنعتی شریف
مدرس: مهدی خرازی

تمرین برنامه نویسی صفرم *
شبکه‌های رایانه‌ای

دانشکده‌ی مهندسی کامپیوتر
پاییز ۱۳۹۵

اهداف تمرین

۱. آشنایی با ساختارهای client-server و peer to peer در قالب ساده

۲. آشنایی با برخی روش‌های رایج برای serialization داده‌ها

۳. آشنایی مقدماتی با برخی روش‌های حفاظت از اطلاعات در شبکه

۴. آشنایی با رمزنگاری اطلاعات و رمزنگاری نامتقارن

۱. مقدمه

هنگام ارسال بسته از یک مبدا به مقصد، عملا هر کسی که به گره‌های میانی در شبکه دسترسی داشته باشد قادر به خواندن محتویات بسته‌ها است. چنانچه اطلاعات مبادله‌شده از اهمیت بالایی برخوردار باشد دسترسی شخص ثالث به آن‌ها می‌تواند مشکل‌ساز باشد. برای مثال یک اپلیکیشن پیام‌رسان را فرض کنید که پیام‌های میان کاربران را به صورت متن ساده [۱] ارسال می‌کند. بنابراین یک مهاجم می‌تواند با تنها نگاه کردن به بسته‌ها از مکالمات خبردار شود.

در هنگام تبادل اطلاعات معمولاً سه شخص درگیر هستند:

- فرستنده که معمولاً به او Alice می‌گوییم.
- گیرنده که معمولاً به او Bob می‌گویم .
- فرد متخاصم که قصد خواندن مکالمات Alice و Bob را دارد و معمولاً به او Eve می‌گوییم.

* با سپاس از سولماز سلیمی، پارسوآ خورسند، پیمان عزتی، رضا میرعسگر شاهی، مهدی بهروزی‌خواه

[۱] plain text

در شرایط روزمره همیشه باید فرض کرد که Eve وجود دارد. به جز در شرایط خاص، مانند تبادل اطلاعات در شبکه داخلی یک سازمان که طکنترل روی زیرساخت شبکه وجود دارد نمی‌توان دسترسی مهاجظم خارجی به اطلاعات را محدود کرد. بنابراین لازم است Alice و Bob اطلاعات خود را رمزنگاری کنند تا فهم آن برای Eve سخت یا ناممکن باشد. اساسی‌ترین سوال در بحث رمزنگاری اینجا مطرح می‌شود: Alice و Bob چگونه می‌توانند هماهنگ کنند که اطلاعات را باید چگونه رمزنگاری کرد؟

از آنجا که Eve به مکالمات آن‌ها دسترسی کامل دارد می‌تواند روش رمزنگاری را از مکالمات قبلی متوجه شود. اگر روش امنی برای تبادل اطلاعات مربوط به روش رمزنگاری داشته باشیم بنابراین می‌توانیم از همان روش برای تبادیل اطلاعات استفاده کنیم! و چنانچه چنین بستر امنی وجود ندارد پس رمزنگاری انجام شده بر اساس آن نیز غیر قابل اعتماد است.

روش پیشنهادی برای حل این مشکل استفاده از رمزنگاری نامتقارن است. هر یک از طرفین قسمتی از اطلاعات لازم برای رمزنگاری [۲] و بازیابی [۳] اطلاعات را در اختیار دارند، بنابراین Eve دیگر نمی‌تواند با خواندن بسته‌های مبادله شده از کلید رمزنگاری با خبر شود. رایج ترین شیوه رمزنگاری نامتقارن رمز نگاری با کلید عمومی [۴] و کلید خصوصی [۵] است.

الگوریتم RSA شناخته‌شده‌ترین شیوه رمزنگاری با کلید عمومی و خصوصی است. این الگوریتم بر این مبنا کار می‌کند که اگر داشته باشیم:

$$\forall m : (m^e)^d \equiv m \mod n$$

آنگاه Alice با فرستادن n و e خود به Bob به او امکان فرستادن پیام‌های رمزنگاری شده را می‌دهد. به دوتایی (e, n) کلید عمومی و به d کلید خصوصی می‌گوییم. به ازای مقادیر بزرگ کلید خصوصی و عمومی (مثلاً اعداد چندصد رقمی) از نظر تئوری پیدا کردن d با داشتن (e, n) بسیار مشکل است و در زمان معقول قابل انجام نیست در صورتی کلید خصوصی و عمومی را می‌توان در زمان کوتاهی تولید کرد.

۲. آشنایی با پرتو

پرتو، سامانه‌ی شبیه‌ساز شبکه‌های کامپیوتری است که در این تمرین به کار گرفته می‌شود. این سامانه به صورت کارخواه‌ـ کارگزار عمل می‌کند. کارگزار پرتو یک توپولوژی شبکه و معماری گره‌ها را برنامه‌ریزی می‌کند و کارخواه‌ها، گره‌های خاص را برنامه‌ریزی می‌کنند. شما می‌توانید با اجرای یک کارخواه به یک گره مجازی که بر روی کارگزار پرتو شبیه‌سازی شده است، متصل شوید. بدین ترتیب کارگزار پرتو بسته‌های شبیه‌سازی شده را به کارخواه‌ها می‌رساند و بسته‌های ارسالی آنها را نیز دریافت کرده و در اختیار گره‌های مجازی قرار می‌دهد. چارچوب کارخواه پرتو، کتابخانه‌ها

[۲] encoding
[۳] decoding
[۴] public key
[۵] private key

و کلاس‌های نرم‌افزاری از پیش نوشته‌شده است که نقش شبیه‌سازی دستگاه‌های شبکه را ایفا می‌کند. کاربر با استفاده از آنها و در محیط آن، توابع، متدها و دیگر نیازهای برنامه‌نویسی خود را تأمین می‌کند. برای آشنایی بیشتر با نحوه‌ی استفاده از چارچوب کارخواه پرتو می‌توانید به مستند راهنمای آن مراجعه کنید.

۳. پروتکل GPG

رمزنگاری اطلاعات تنها در صورتی مفید است که استفاده همه‌گیر داشته باشد. GPG ساختاری متن باز [۶] برای به اشتراک گزاری کلیدهای عمومی است تا بتوان به واسطه آن به راحتی ارتباطات را رمزنگاری کرد. از رایج‌ترین کاربردهای GPG ارسال ایمیل‌های رمزنگاری شده است. در این پروتکل تعدادی کارگزار شناخته‌شده برای به اشتراک گذاری کلیدهای عمومی وجود دارد که یک نگاشت از آدرس ایمیل به کلید عمومی را نگهداری می کنند . برای ارسال ایمیل رمزنگاری شده کافی است کلید عمومی صاحب آدرس مقصد از یکی از کارگزارها دریافت کنیم و سپس با استفاده از کلید خصوصی اطلاعات را رمزنگاری کنیم. این قابلیت معمولاً در قالب افزونه plugin برای نرم‌افزارهای مختلف ارائه می‌شود. در این تمرین قصد داریم ساختار مشابهی را روی سامانه پرتو پیاده‌سازی کنیم.

۴. ساختار شبکه

توپولوژی شبکه در این تمرین یک گراف کامل است. همه گره‌های این گراف به هم متصل هستند و هر گره با یک شاخص [۷] عددی مشخص می‌شود. گره با اندیس صفر به عنوان کارگزار GPG عمل می‌کند. هر گره پس از اجرا به درخواست کاربر جفت کلید خصوصی و عمومی خود را تولید می‌کند و کلید عمومی را به کارگزار ارسال می‌کند. کارگزار یک نگاشت از شاخص به کلید عمومی نگهداری می‌کند. حال هر گره می‌تواند از طریق خط فرمان دستورهایی برای فرستادن اطلاعات به سایر گره‌ها دریافت کند. پیش از ارسال اطلاعات به یک گره می‌بایست کلید عمومی آن از کارگزار دریافت شود و سپس اطلاعات رمزنگاری شده به آن ارسال شود.

با توجه به ساختار گراف کامل شبکه، هر گره برای ارسال بسته تنها نیاز به دریافت کلید عمومی مورد نظر از کارگزار مرکزی و ارسال بسته روی لینک ارتباطی با مقصد است.

لازم به ذکر است که چنین ساختاری علی رغم سادگی در شرایط واقعی در سطح اینترنت قابل استفاده نیست اما می‌تواند مدلی ساده برای مدیریت ارتباطات در یک شبکه کوچک باشد. در پیاده‌سازی‌های تجاری ارسال بسته‌ها معمولاً از طریق ساختارهای پیچیده peer to peer و مسیریابی عادی IP صورت می‌گیرد.

---

[۶] open source

[۷] index

۵. پیاده سازی

کدی که شما می‌نویسید عملکرد یک گره از ساختار گفته شده را پیاده‌سازی می‌کند. این گره ممکن است یکی از کاربرها و یا کارگزار اشتراک‌گذاری کلیدهای باشد، بنابراین عملکرد کد شما وابسته به شاخص گره متفاوت خواهد بود.

۱.۵. ملاحظات کلی

برنامه شما روی سامانه پرتو اجرا می‌شود. هر گره از شبکه پس از اجرا به پرتو متصل شده و بسته‌های خود را به آن می‌فرستد. کارگزار پرتو با توجه به بسته دریافتی و توپولوژی مشخص شده برای شبکه، آن را به گره مقصد و روی رایانه شما بازمی‌گرداند.

هر گره تعدادی درگاه دارد که مستقیم به گره‌های دیگر متصل هستند. چنانچه گره‌ای بخواهد بسته‌ای را به گره i ام ارسال کند آن را روی درگاه i می‌فرستد.

پروتکل این تمرین بر روی پروتکل UDP پیاده‌سازی می‌شود بنابراین برای ارسال بسته نیاز به پر کرد اطلاعات مربوط به لایه‌های Ethernet ، IP و UDP نیز وجود دارد. کار با این لایه‌ها و پروتکل‌های مربوط به آن‌ها به تفصیل در ادامه مباحث درس و تمرین‌ها آینده پوشش داده می‌شود، بنابراین در این تمرین محتویات سرآیند این لایه‌ها توسط سامانه پرتو برای شما مقداردهی می‌شود.

۲.۵. استاندارد JSON

محتوای منتقل شده بین گره‌ها در قالب استاندارد JSON [۸] ارسال می‌شود. JSON داده ساختار بازگشتی و ساده‌ایست که برای انتقال اطلاعات در قالب مناسب برای استفاده انسان کاربرد دارد. این استاندارد پرکاربردترین روش بسته‌بندی [۹] داده در سطح وب است.

داده ساختار JSON متشکل از تعدادی object است که هرکدام از مجموعه از جفت‌های کلید مقدار [۱۰] تشکیل شده‌اند. کلید همواره یک رشته متشکل‌از کاراکترهای استاندارد است و مقدار نیز می‌تواند خود یک مقدار عددی، رشته، مقدار منطقی، object جدید و یا یک آرایه از object ها باشد. این ساختار همواره در بالاترین سطح متشکل از یک object است.

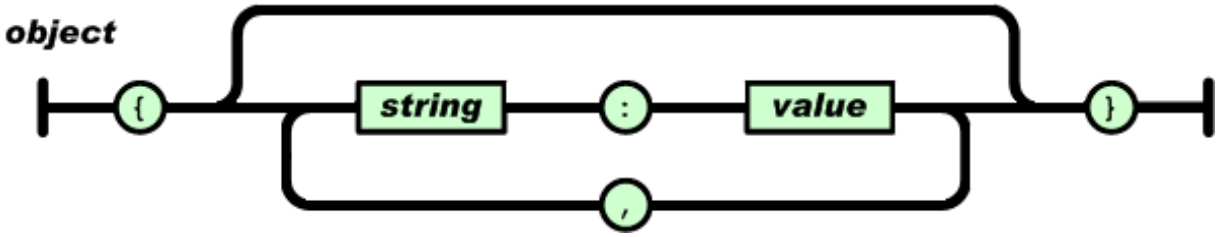
شکل زیر ساختار یک object را نشان می‌دهد:

---

[۸] JavaScript Object Notation

[۹] serialization

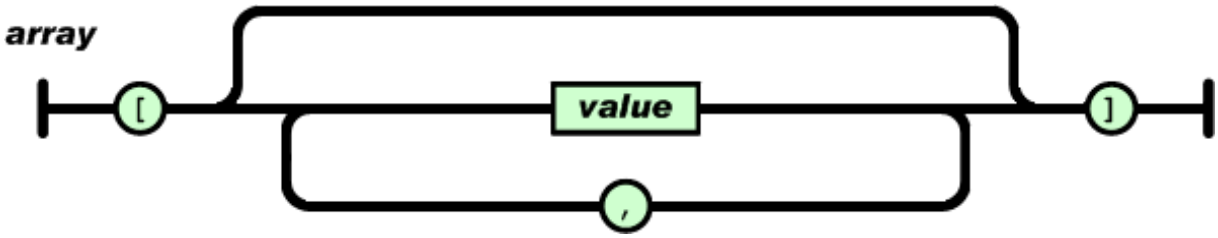
[۱۰] key-value pair



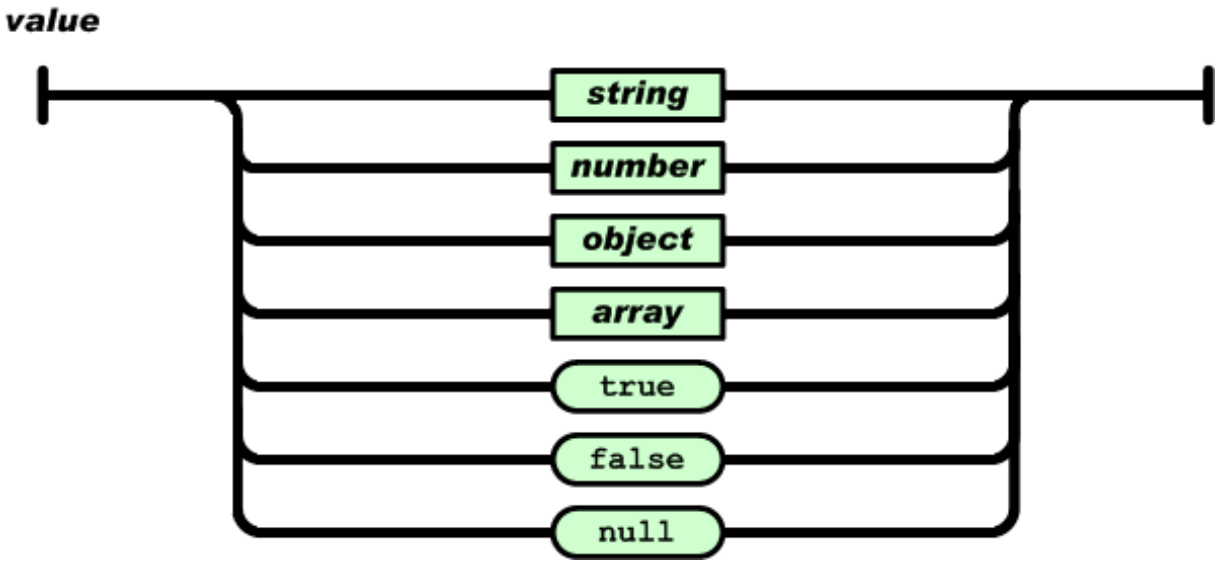
به عنوان نمونه:

```
{
  "first_name" : "Parsoa",
  "last_name" : "Khorsand",
  "alive" : "false"
}
```

یک آرایه نیز ساختار زیر را دارد:



همانطور که گفته شده value می‌تواند خود یک آرایه یا object جدید باشد، بنابراین ساختار value نیز به شکل زیر است:



در این تمرین تنها به ساختارهای ساده شامل یک object بسنده می‌کنیم.

برای آشنایی بیشتر با JSON می‌توانید وبگاه کارگروه توسعه آن را مطالعه کنید.

۳.۵ ساختار بسته‌ها

بسته‌ها ساختار ساده‌ای دارند که متشکل از یک سرآیند و یک محتوا [۱۱] است. سرآیند مشخص کننده نوع بسته و محتوا نیز اطلاعات ردوبدل شده است. سرآیند بسته‌های به شکل زیر است:

type	op	index	status
length			

فیلد type تک بایتی نشان‌دهنده نوع بسته است که می‌تواند یک درخواست [۱۲] و یا یک پاسخ [۱۳] باشد. مقدار صفر نشان‌دهنده درخواست و مقدار یک نشان‌دهنده پاسخ است.

فیلد تک بیتی op نشان‌دهنده نوع عملیات است که می‌تواند فرستادن پیام و یا ارسال کلید باشد. مقدار op برای بسته‌های مربوط به ارسال و دریافت پیام برابر یک و برای بسته‌های مربوط به انتشار کلید برابر صفر است.

فیلد ۴ بیتی index نیز مشخص کننده شماره گره‌ای از شبکه است که عملیات این بسته مربوط به آن است. در بسته‌های مربوط به ارسال و دریافت پیام این فیلد شماره گره مقصد را مشخص می‌کند و در بسته‌های مربوط به دریافت کلید نیز شماره گره مورد نظر را تعیین می‌کند. همچنین این مقدار در درخواست‌های مربوط به انتشار کلید صفر قرار داده می‌شود.

در نهایت فیلد status نیز نشان‌دهنده وضعیت درخواست ارسال شده است. مقدار این فیلد دو بیتی در بسته‌های از نوع درخواست برابر صفر و در بسته‌های از نوع پاسخ نشان‌دهنده وضعیت پردزاش درخواست است.

در ادامه فیلد ۸بیتی length نیز طول محتوای بسته را مشخص می‌کند.

پس درنهایت سرآیند این پروتکل دو بایت (یا همان ۱۶ بیت) است.

۴.۵ کارگزار مرکزی

این گره از شبکه که با اندیس صفر مشخص می‌شود وظیفه نگهداری و اشتراک گذاری کلیدهای عمومی را بر عهده دارد. آدرس IP این گره همواره 192.168.128.0 خواهد بود .

کارگزار درخواست‌ها برای ثبت کلیدهای عمومی را در قالب درخواست‌هایی با ساختار زیر دریافت می‌کند:

0	0	0000	00
length			

<pre>{
"e": 7,
"n": 1025
}</pre>			

[۱۱] payload
[۱۲] request
[۱۳] response

باقی سرآیند این بسته مطابق با توضیحات قسمت قبل پر خواهد شد.

مشاهده می‌شود که body یک json با دو کلید e و n که کلید عمومی را مشخص می‌کنند.

در پاسخ به چنین درخواست‌هایی، چنانچه ساختار محتوا مطابق شکل فوق باشد کارگزار باید یک response بدون محتوا با مقدار status دودویی 11 (یعنی خوب پیش رفتن عملیات) و در غیر این صورت با status 01 بازگرداند. با توجه به خالی بودن payload مقدار length در سرآیند برابر صفر خواهد بود.

1	0	0000	11
00000000			

یا

1	0	0000	01
00000000			

دقت کنید که در تمام پاسخ‌های بازگردانده شده توسط کارگزار مقدار index باید برابر مقدار موجود در درخواست مربوطه باشد.

۵.۵. گره‌های کاربر

هر گره کاربر قبل از شروع مکالمه با سایر کابران باید کلید عمومی و خصوصی خود را تولید کنید. این‌کار زمانی انجام می‌شود که فرمان زیر در خط فرمان دریافت شود:

```
generate
```

با دریافت این فرمان باید ابتدا پیام زیر چاپ شود:

```
generating key pair ...
```

پس از پایان فرآیند تولید کلیدها باید خروجی زیر چاپ شود:

```
keys generated!
public: (e,n)
private: d
```

پس از فرمان `generate` با دریافت فرمان `publish` باید کلید عمومی به کارگزار ارسال شود. در هنگام آغاز ارسال این فرآیند باید خروجی زیر چاپ شود:

```
publishing public key ...
```

چنانچه ارسال کلیدهای عمومی با دریافت کد ⌧⌧ همراه باشد باید پیام زیر چاپ شود:

```
successfully published public key
```

در غیر این صورت:

```
public key publication failed
```

چنانچه قبل از اجرای `generate` دستور `publish` دریافت شود، گره باید ابتدا پیام زیر را چاپ کند:

```
no keys found
```

سپس خودش دستور `publish` را اجرا کند.

گره‌های کاربر می‌توانند از طریق خط فرمان دستوراتی مبنی بر ارسال پیام به سایر گره‌ها دریافت کنند. این فرمان حالت کلی زیر را دارند:

```
send <index> <message>
```

در این ساختار، <index> اندیس گره مقصد و <message> پیام ارسال شده است. به طور مثال:

```
send 4 hello there, are you ok for lunch today at 2?
```

با دریافت چنان دستوری باید ابتدا کلید عمومی مقصد از کارگزار دریافت شود. به این منظور کافی است یک درخواست مشابه زیر به کارگزار ارسال شود:

0	0	index	00
00000000			

که مطابق قبل، <index> اندیس گره مورد نظر است. در صورتی که <index> معتبر باشد کارگزار یک پاسخ با ساختار زیر برمی‌گرداند:

0	0	index	11
length			

<pre>{
"e": 7,
"n": 1025
}</pre>			

چنانچه اطلاعات اندیس مورد نظر روی کارگزار موجود نباشد یک پاسخ با status برابر 10 بازگردانده می‌شود:

1	0	index	10
00000000			

در صورت دریافت خطای فوق باید پیام زیر در خروجی چاپ شود:

```
target client not found!
```

برای ارسال پیام به مقصد باید یک درخواست با ساختار زیر ارسال شود:

0	1	index	00
length			
<message>			

دقت کنید که در این ساختار <index> اندیس گره ارسال کننده پیام است.

با دریافت پیام توسط یک گره باید خروجی زیر چاپ شود:

```
message received from <index>:
<message>
----
```

۶.۵ تولید جفت کلید

در پیاده‌سازی‌های عملی ابتدا e را تولید و سپس از روی آن n و d را محاسبه می‌کنیم. برای e معمولاً از اعداد فرما که به شکل $۲^{۲^x} + ۱$ هستند استفاده می‌شود. زمانی تصور می‌شد که هر عدد به این شکل حتما اول است. این اعداد به سرعت به سمت مقادیر بسیار بزرگ رشد می‌کنند به همین دلیل تا قبل از اختراع کامپیوتر تنها اول بودن ۵ عدد اول بررسی شده بود. نشان داده شده‌است که تنها همین ۵ عدد اول هستند و سایر اعداد فرما اول نیستند. در محاسبات عملی معمولا از $e = ۶۵۵۳۷$ استفاده می‌شود. در این‌جا برای سادگی از $e = ۱۷ = ۲^{۲^۲} + ۱$ استفاده می‌کنیم. حال از الگوریتم زیر برای تولید کلیدها استفاده می‌کنیم:

Input: k, length of the key

Result: (N, e, d)

set $e = 17$;

while $p \not\equiv 1 \mod e$ **do**

  $p \leftarrow genPrime(k/2)$;

end

while $q \not\equiv 1 \mod e$ **do**

  $q \leftarrow genPrime(k - k/2)$;

end

$N \leftarrow pq$;

$L \leftarrow (p-1)(q-1)$;

$d \leftarrow modInv(e, L)$;

در الگوریتم فوق تابع $generatePrime(a)$ یک عدد اول با نمایش a بیتی (یعنی بیت a ام برابر یک) باز می‌گرداند. در عمل طول کلید را 1024 یا بیشتر قرار می‌دهند اما در این تمرین برای کاهش زمان محاسبه و سادگی از طول کلید 12 استفاده می‌کنیم. برای هماهنگی میان پیاده‌سازی شما و سامانه نمره‌دهی توابع لازم برای تولید کلیدها در اختیار شما قرار داده می‌شود.

تابع $modInv$ مقدار d را بازمی‌گرداند که $ed \equiv 1 \mod L$. کد مورد نیاز برای محاسبه این مقدار در قالب تابع $modularInverse(e, L)$ در دسترس است.

۷.۵. رمزنگاری

برای رمزنگاری پیام از روش زیر استفاده می‌کنیم:

۱. کلید عمومی مقصد را دریافت کن

۲. پیام را به قسمت‌های ۸ بایت (یک کارکتری) تقسیم کنیم

۳. برای هر قسمت c از پیام مقدار $c^e \mod n$ را محاسبه کن و به جای آن قرار بده.

به صورت استاندارد هر کاراکتر با ۸ بیت نمایشه داده می‌شود اما مقادیر تولید شده با روش فوق ممکن است بیشتر از ۸ بیت فضا لازم داشته باشند (با توجه به مقدار n) بنابراین پس از رمزنگاری هر کاراکتر تعداد بیت‌های مورد نیاز برای نمایش رمزنگاری شده آن را به نزدیک‌ترین مضرب ۸ گرد می‌کنیم و ضریب حاصل (تعداد بیت‌های مورد نیاز گرد شده تقسیم بر ۸) را قبل از مقدار حاصل در رشته پیام قرار می‌دهیم. به طور مثال اگر پیام ارسال شده رشته hi باشد و کاراکتر h بعد از رمزنگاری به ۱۸ بیت فضا و کاراکتر i بعد از رمزنگاری به ۶ بیت فضا نیاز داشته باشد رشته حاصل به این شکل می‌شود:

```
3<byte><byte><byte>1<byte>
```

در ادامه برای بازگشایی رمز از روش زیر استفاده می‌کنیم.

۱. مقادیر رمزنگاری شده را با توجه به ضرایب موجود در پیام جداسازی کن

۲. به ازای هر قسمت c از پیام مقدار $c^d \mod n$ را قرار بده

با توجه به اینکه به ازای هر m داریم $(m^e)^d \equiv m \mod n$ بنابراین روش فوق به درستی کار می‌کند.

نکات ضروری

- از آنجا که نمره‌دهی تمرین به صورت خودکار انجام می‌شود جزییات مطرح شده حین پیاده‌سازی باید به طور کامل رعایت شود.
- کلاس حل تمرین برای توضیح نحوه کار با پروتو و اصول نوشتن کد برای آن برگزار خواهد شد.
- سوالات و ابهامات خود در مورد تمرین را در پست‌های مشخص شده در پیاتزا مطرح کنید.
- پیاده‌سازی کد باید توسط خود دانشجو صورت گرفته باشد. شباهت غیرقابل توجیح میان کدها و یا استفاده مستقیم از کدهای موجود در اینترنت حتی با ذکر منبع به مثابه تقلب است.