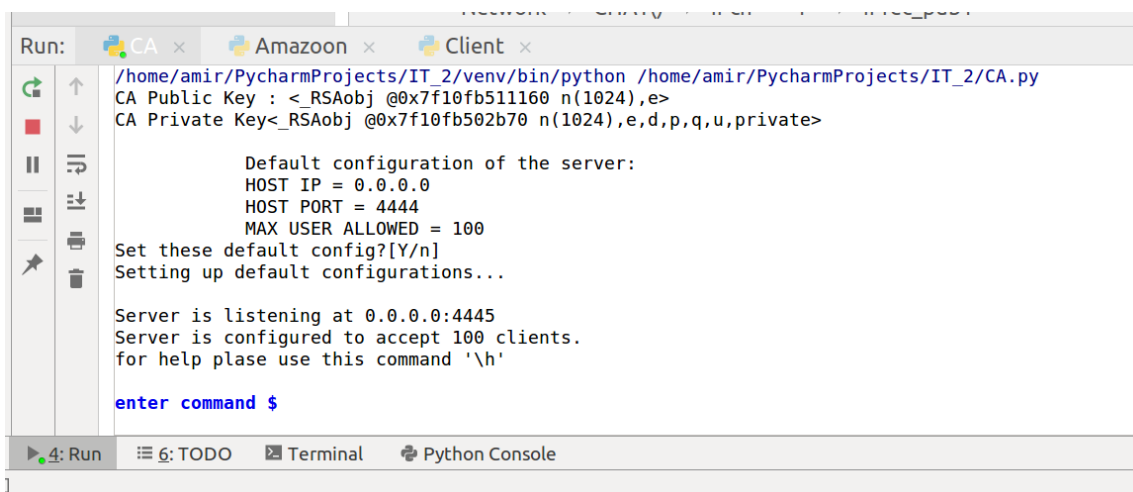


سوال ۱)

در این سوال من سرور های CA ، Amazon و Client را کد نویسی کرده ام .

برای راه اندازی ابتدا شما باید CA را راه بیندازید ، برنامه با پایتون ۳ نوشته شده و با Pycharm کد زده شده است .



```
Run: CA x Amazon x Client x
/home/amir/PycharmProjects/IT_2/venv/bin/python /home/amir/PycharmProjects/IT_2/CA.py
CA Public Key : < RSAObj @0x7f10fb511160 n(1024),e>
CA Private Key<_RSAObj @0x7f10fb502b70 n(1024),e,d,p,q,u,private>

Default configuration of the server:
HOST IP = 0.0.0.0
HOST PORT = 4444
MAX USER ALLOWED = 100
Set these default config?[Y/n]
Setting up default configurations...

Server is listening at 0.0.0.0:4445
Server is configured to accept 100 clients.
for help please use this command '\h'

enter command $
```

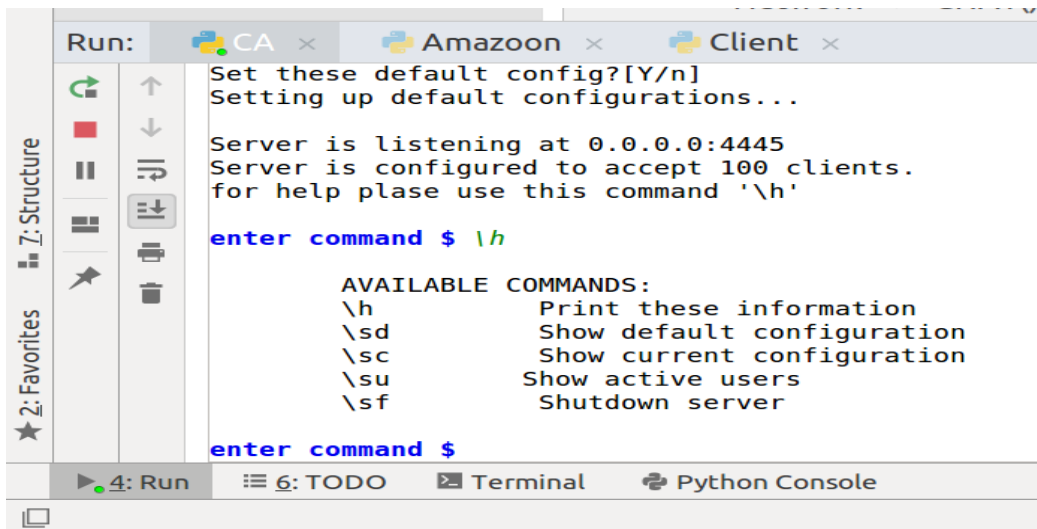
بله همونجور که در تصویر مشخص است در ابتدای کار کلید عمومی و خصوص سرور چاپ می شود که تا پایان برنامه همین مقدار خواهد بود .

سپس کانفیگ دیفالت برنامه که در آن شماره آی پی و پورت و ماکسیمم یوزری که می تواند CA پذیرش کد نمایش داده می شود

و در صورتی که بخواهید می توانید با زدن کلید n آن را تغییر دهید در غیر این صورت Enter یا Y می زنیم که با همین تنظیمات سرور راه اندازی شود .

سرور با همین تنظیمات بر روی پورت ۴۴۴۵ راه اندازی می شود و ماکسیمم به ۱۰۰ نفر Certificate می دهد .

با استفاده از کامند /h می توانیم برخی از دستوراتی که این سرور می گیرد را ببینیم .



```
Run: CA x Amazoon x Client x
Set these default config?[Y/n]
Setting up default configurations...

Server is listening at 0.0.0.0:4445
Server is configured to accept 100 clients.
for help please use this command '\h'

enter command $ \h

AVAILABLE COMMANDS:
\h          Print these information
\sd         Show default configuration
\sc         Show current configuration
\su         Show active users
\sfc        Shutdown server

enter command $
```

نشان دادن دوباره تنظیمات پیش فرض و نشان دادن تنظیمات فعال ، نشان دادن اسم یوزر هایی که به آن ها Certificate اعطا شده و خاموش کردن سرور از جمله کامند هایی است که می توان با توجه به راهنمایی فوق به سرور داد .

- CA server support more than one connection (thread programming) :

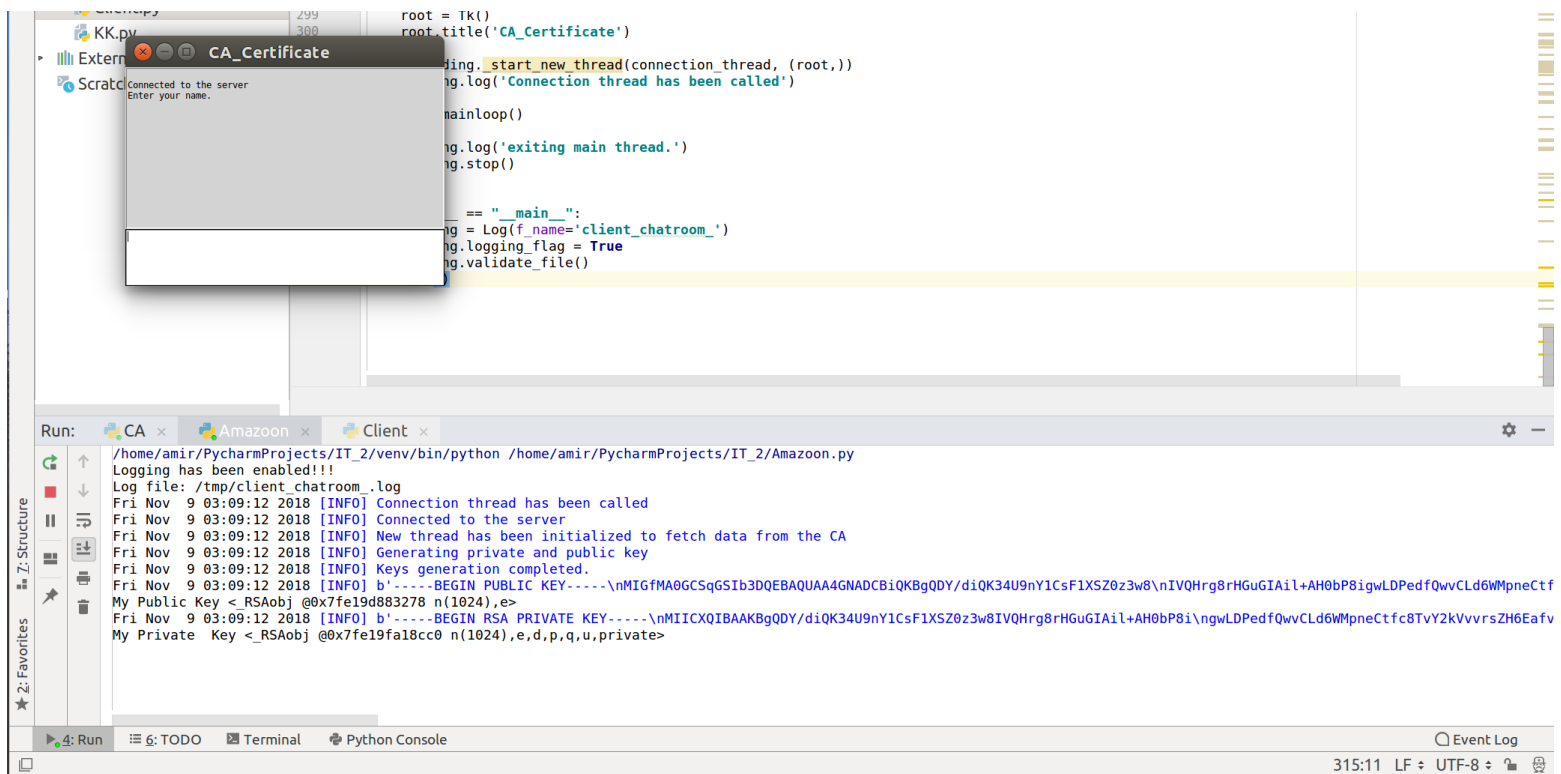
```
threading._start_new_thread(cli_obj.run, (""))
```

```
thread_cli = threading.Thread(target=self.init_clients, args=())
```

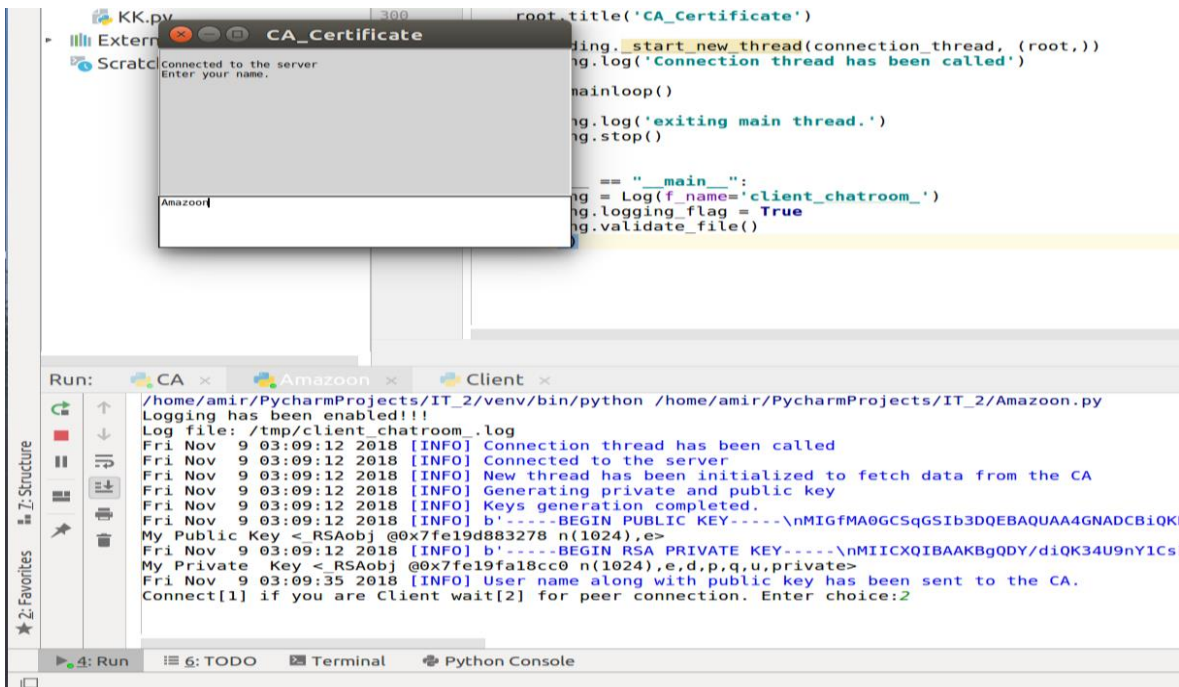
برنامه Client و Amazoon برای جلوگیری از خطا و دنبال کردن پروسه ها در یک فایل نوشته شده که شما کافی است از آن دوبار کپی بگیرید یک بار با نام Amazoon.py و یکبار با نام Client.py در حین اجرای برنامه از شما سوال می شود که کلاینت یا سرور هستید

که کلا مشکلی نیست برای جدا کردن این دو ولی این گونه برنامه را بهتر درک می کنیم .

(برای جدا کردن فقط کافی است در یکی 'ch='۱' و در دیگری '۲' ست شود که پرسشی پرسیده نشود !! به همین سادگی)



آمازون که سرور می باشد را ران می کنیم . لاگ اتفاقات در فایل در /tmp ریخته می شود .
 کلید ها تولید شده و محض اطلاع به شما نمایش داده می شود . سپس کادری باز شده که به
 CA وصل شده و نام شما را برای تولید Certificate می پرسد .
 Amazon می نویسیم ، نام شما همراه با کلید عمومی شما به CA برای امضا شدن فرستاده
 می شود .



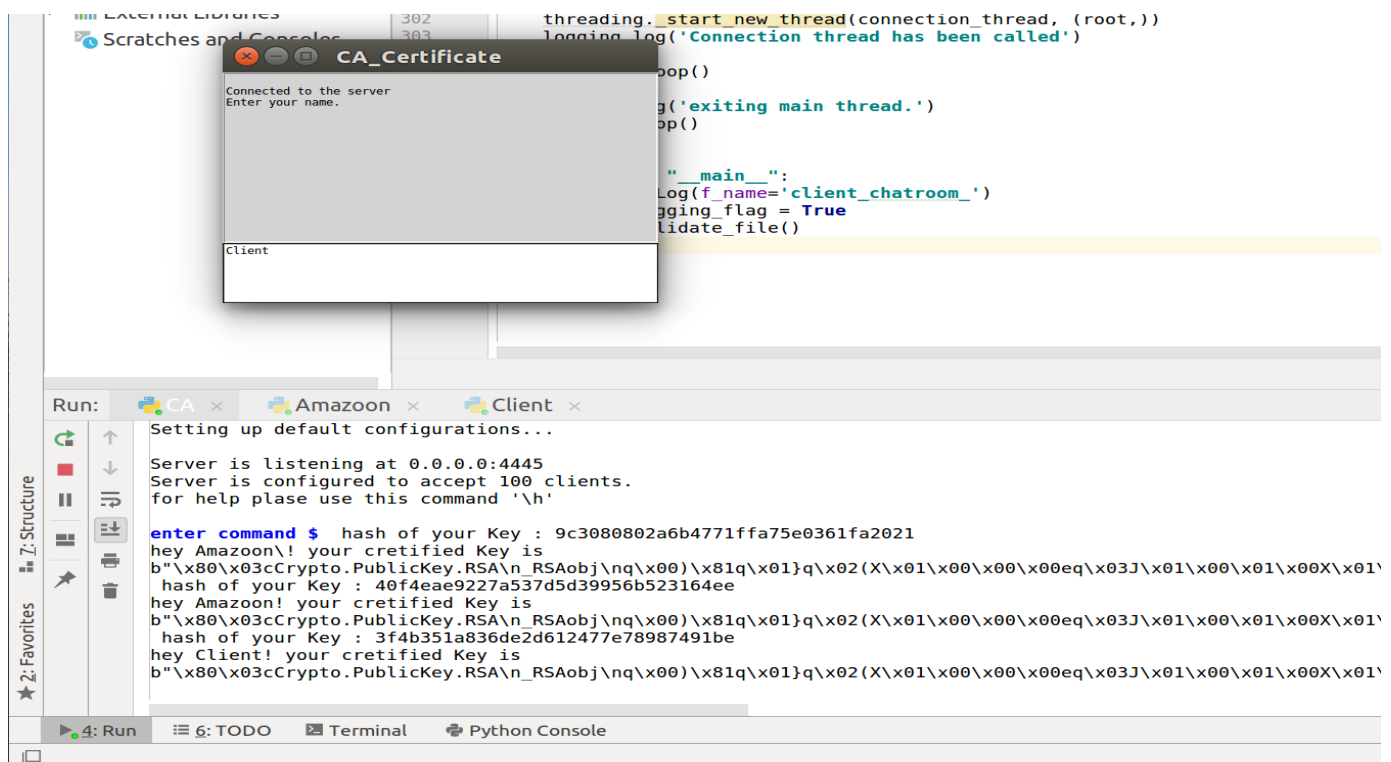
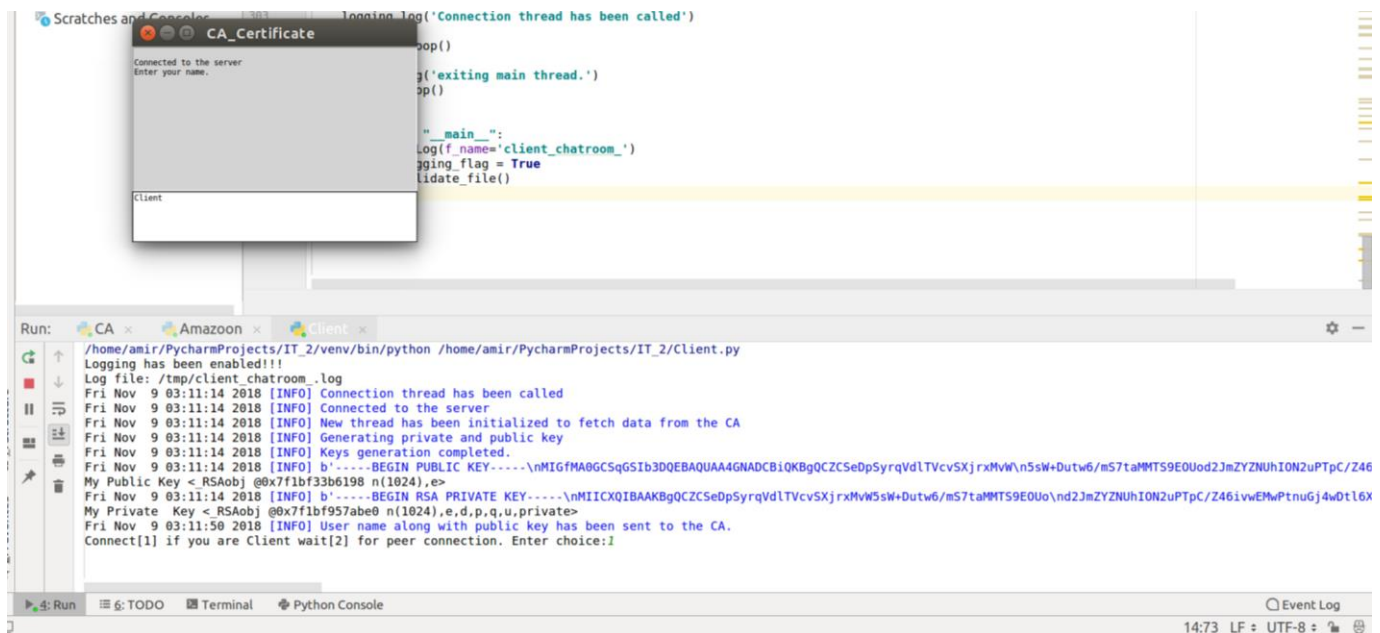
سپس از شما پرسیده می شود که سرور هستید یا کلاینت که ۲ را می نویسم اما هنوز Enter نزده ام .

[illegible]

Hash کلید آن گرفته شده است و نمایش داده شده و سپس آن را با کلید پرایویت خود امضا کرده و به شما فرستاده است .

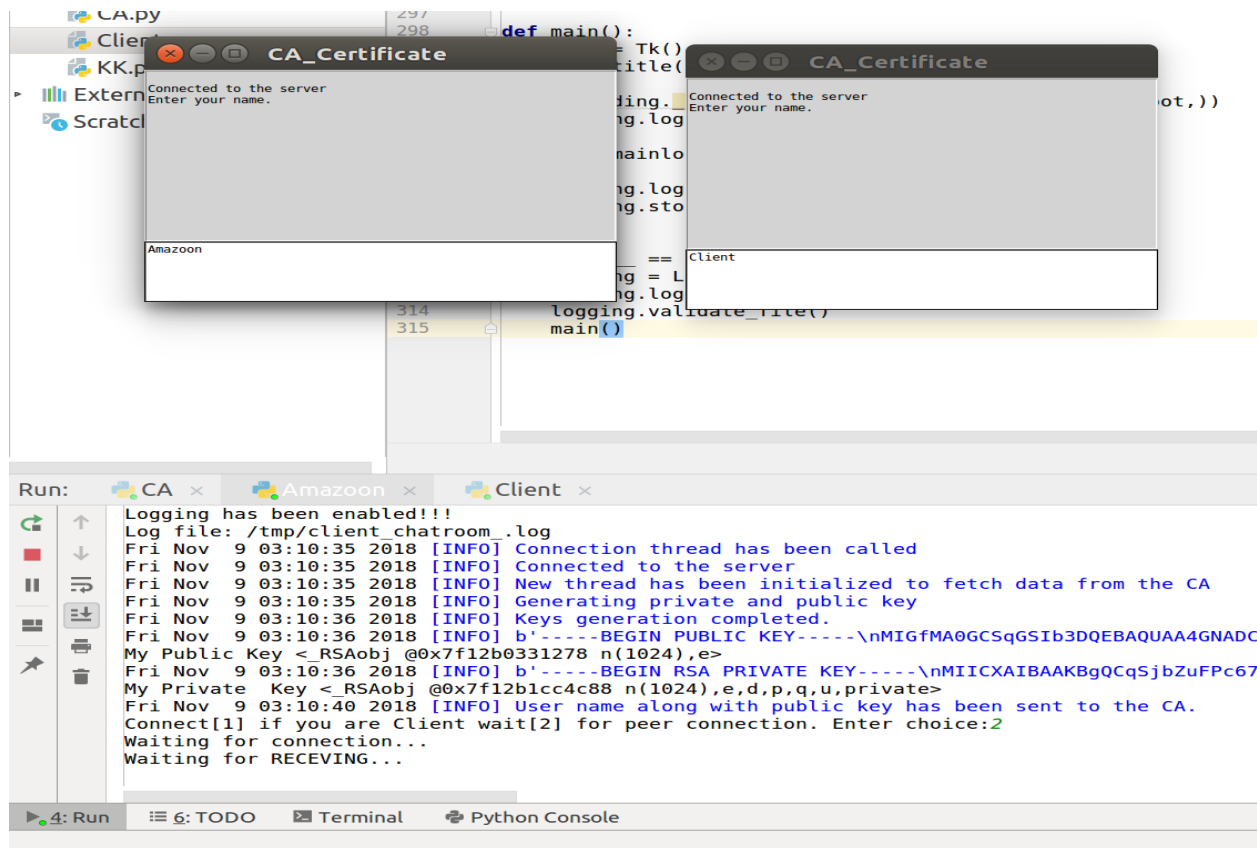
پس شما الان امضا شده هش کلید خودتون را با کلید پرایویت CA و کلید عمومی CA دارید .

همین اتفاق مشابه برای Client تکرار می شود که من عکس های آن را می گذارم



گفتیم که پس از وارد کردن ۱ و ۲ هنوز Enter را نزده بودیم خب میریم اول از سرور را
میزنیم و بعد از کلاینت .

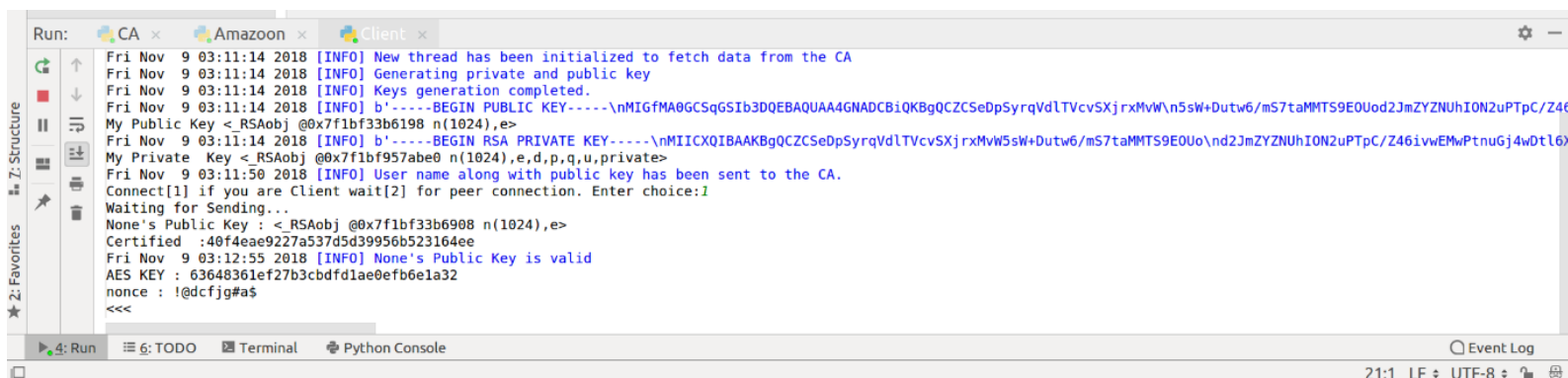
وقتی از سرور را میزنیم میره توی حالت این که کسی بهش وصل بشه و وقتی میریم از کلاینت را میزنیم ، به سرور متصل میشه .



در این حین هر دو سرور کلید عمومی خود و سرتیفیکیت را برای هم می فرستند .

سرور یک KEY برای رمز AES تولید می کند و آن را با پابلیک Key کلاینت رمز کرده و برای آن می فرستد .

کلاینت Ensharedkey که رمز شده کلید می باشد را دریافت کرده و با کلید خصوصی خود باز می کند .



جدای از این که یک KEY برای این ارتباط صادر شده و رمز شده و استفاده خواهد شد .

یک عبارت nonce تصادفی هم تولید می شود و برای استفاده در AES استفاده خواهد شد .

پروتکل ما برای استفاده از این nonce در کنار کلید قرار دادن اون و گرفتن Hash آن به عنوان کلید رمزنگاری می باشد .

این گونه هر وقت nonce تغییر کرد می توان بدون این که دوباره نیازی به رد و بدل کلید AES باشد رمز عوض شود .

nonce به صورت فاش فرستاده می شود .

که در تصویر بالا می تونید nonce دریافت شده ، کلید رمز AES باز شده ، کلید عمومی آمازون و سرتیفیکت آن را ببینید .

در log بالا نوشته شده است که Public key is valid و این به منظور چک کردن سرتیفیکت با public key فرستاده شده می باشد . که موفقیت آمیز بوده

چت آغاز شده ! و در تمام مدت چت ما متن رمز شده را همراه با Hash پیام رمز نشده آن به منظور برقراری integrity می فرستیم .

```

Run: CA x Amazon x Client x
Fri Nov 9 03:10:36 2018 [INFO] Keys generation completed.
Fri Nov 9 03:10:36 2018 [INFO] b'-----BEGIN PUBLIC KEY-----\nMIGfMA0GCs5qGSIb3DQEBAQUAA4GNADCBiQKBgQCq5jbZuFPc67oNxDUyUQhcBWun\Fd2NMcaGc
My Public Key < RSAobj @0x7f12b0331278 n(1024),e>
Fri Nov 9 03:10:36 2018 [INFO] b'-----BEGIN RSA PRIVATE KEY-----\nMIICXAIBAAKBgQCq5jbZuFPc67oNxDUyUQhcBWun\Fd2NMcaGQ8L2GT1S81evTXX\mn9sjT
My Private Key < RSAobj @0x7f12b1cc4c88 n(1024),e,d,p,q,u,private>
Fri Nov 9 03:10:40 2018 [INFO] User name along with public key has been sent to the CA.
Connect[1] if you are Client wait[2] for peer connection. Enter choice:2
Waiting for connection...
Waiting for RECEIVING...
Fri Nov 9 03:12:55 2018 [INFO] public key is valid
shared_key63648361ef27b3cbdfdaefb6e1a32<class 'str'>
customor_pub_key : < RSAobj @0x7f12b03316a0 n(1024),e><class 'Crypto.PublicKey.RSA._RSAobj'>
KEYYYYYYYYYYYYYY63648361ef27b3cbdfdaefb6e1a32 type : <class 'str'>
customor_pub_key : < RSAobj @0x7f12b03316a0 n(1024),e> 3f4b351a836de2d612477e78987491be
<<< Hi my name is Amazon
<<<

Run: CA x Amazon x Client x
My Public Key < RSAobj @0x7f1bf33b6198 n(1024),e>
Fri Nov 9 03:11:14 2018 [INFO] b'-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQCZCSeDpSyrqVdLTVcvSXjrxMvW5Sw+Dutw6/ms7taMMT
My Private Key < RSAobj @0x7f1bf957abe0 n(1024),e,d,p,q,u,private>
Fri Nov 9 03:11:50 2018 [INFO] User name along with public key has been sent to the CA.
Connect[1] if you are Client wait[2] for peer connection. Enter choice:1
Waiting for Sending...
None's Public Key : < RSAobj @0x7f1bf33b6908 n(1024),e>
Certified : 40f4eae9227a537d5d39956b523164ee
Fri Nov 9 03:12:55 2018 [INFO] None's Public Key is valid
AES KEY : 63648361ef27b3cbdfdaefb6e1a32
nonce : !@dcfjg#a$
<<< Encrypted message : b'72KThn/WVg9lGhJTUbSEyVTtb/vOCTY7nHTCro9lBis='
Delivered Hash of message for check integrity 89ccc7808ff0c6622bbb2764a5d0c620
Fri Nov 9 03:13:26 2018 [INFO] We Have integrity
>>> Hi my name is Amazon
<<<
  
```

```
Run: CA x Amazon x Client x
Fri Nov 9 03:11:14 2018 [INFO] b'-----BEGIN RSA PRIVATE KEY-----\nMIICXQIBAAKBgQCZCSeDpSyrqVd1TVcvSXjrxMvW!\nMy Private Key < RSAobj @0x7f1bf957abe0 n(1024),e,d,p,q,u,private>\nFri Nov 9 03:11:50 2018 [INFO] User name along with public key has been sent to the CA.\nConnect[1] if you are Client wait[2] for peer connection. Enter choice:1\nWaiting for Sending...\nNone's Public Key : < RSAobj @0x7f1bf33b6908 n(1024),e>\nCertified :40f4eae9227a537d5d39956b523164ee\nFri Nov 9 03:12:55 2018 [INFO] None's Public Key is valid\nAES KEY : 63648361ef27b3cbdfdlae0efb6e1a32\nnonce : !@dcfjg#a$\n<<< Encrypted message : b'72KThn/WVg9lGhJTUbSEyVTtb/vOCTY7nHTCro9lBis='\nDelivered Hash of message for check integrity 89ccc7808ff0c6622bbb2764a5d0c620\nFri Nov 9 03:13:26 2018 [INFO] We Have integrity\n>>> Hi my name is Amazon\n<<< Im glad to buy from you\n<<<
```

```
Run: CA x Amazon x Client x\nMy Private Key < RSAobj @0x7f12b1cc4c88 n(1024),e,d,p,q,u,private>\nFri Nov 9 03:10:40 2018 [INFO] User name along with public key has been sent to the CA.\nConnect[1] if you are Client wait[2] for peer connection. Enter choice:2\nWaiting for connection...\nWaiting for RECEIVING...\nFri Nov 9 03:12:55 2018 [INFO] public key is valid\nshared_key63648361ef27b3cbdfdlae0efb6e1a32<class 'str'>\ncustomer_pub_key : < RSAobj @0x7f12b03316a0 n(1024),e><class 'Crypto.PublicKey.RSA._RSAobj'>\nKEYYYYYYYYYYYYYY63648361ef27b3cbdfdlae0efb6e1a32 type : <class 'str'>\ncustomer_pub_key : < RSAobj @0x7f12b03316a0 n(1024),e> 3f4b351a836de2d612477e78987491be\n<<< Hi my name is Amazon\n<<< Encrypted message : b'dq9sA90cZwu/r9+W27V0BKNHu3RPGZBk0QDpbqivnYo='\ndeliver hash 679b2ed163e362244fe328d928fef01f\nFri Nov 9 03:14:22 2018 [INFO] We Have integrity\n>>> Im glad to buy from you\n<<<
```

بخشی از چت را در ۴ عکس بالا دیدیم .

مشاهده یوزر های فعال در CA و خاموش کردن سرور CA :

```
Run: CA x Amazon x Client x\n\n\\sd Show default configuration\n\\sc Show current configuration\n\\su Show active users\n\\sf Shutdown server\n\nenter command $ \\su\n['Client', 'Amazon']\n\nenter command $ \\sf\nWARNING: All users will be disconnected with out any notification!!\nDo you really want to close server?[Y/N] Y\nShuting down server...\nServer has stopped listening on opened socket.\n\nProcess finished with exit code 0
```


گرفتن نام و کلید عمومی بقیه و تولید کلید امضا شده به همراه کلید عمومی CA

```
def run(self, *args) :
    data = self.conn.recv(4000)

    if data :
        self.userName, self.PUBLIC_KEY = pickle.loads(data)

    if self.PUBLIC_KEY :
        CertifiedKey = self.certifying_client_key()

        CertifiedKey2 = (CA.CA_pub_key, CertifiedKey)
        CertifiedKey2 = pickle.dumps(CertifiedKey2)
        self.conn.send(CertifiedKey2)
        print("hey " + str(self.userName) + "! your cretified Key is \n" + str(CertifiedKey2))
```

اگر کلاینت و سرور نتوانستند به CA وصل شوند تا چندین بار Retry می کنند .

```
except Exception as e :
    msg = "[Retry {}] {}".format(retry_count + 1, e)
    logging.log(msg)
    retry_count += 1
    if retry_count == 1 :
        gui = GUI(root, None)
        gui.update("Failed to connect the server.\n" + \
            "Started retrying.")
        gui.update("Retry connecting...")
        time.sleep(5)
        gui_flag = True
    elif 4 > retry_count:
        time.sleep(5)
        gui_flag = True
    elif retry_count == 5 :
```

در این تابع کلید عمومی CA و کلید امضا شده خودمان را دریافت میکنیم و سپس تابع Chat را برای ادامه کار فرا می خونیم .

```
def initEncryption(self, userName) :
    global KEY
    msg_send = (userName, self.pub_key)
    msg_send = pickle.dumps(msg_send)
    self.client.send(msg_send)
    logging.log("User name along with public key has been sent to the CA.")
    data = self.client.recv(4000)
    if data:
        self.ca_pub_key, self.certified_pub_key = pickle.loads(data)
        self.CHAT(userName)
```

تابع CHAT مهمریت تابع می باشد که پروسه تولید کلید AES و احراز هویت با صدا زدن تابع Check_Ca محقق می شود .

در اینجا آمازون و سرور به یکدیگر شناخته می شوند و با استفاده از تابع ودستور زیر کلید و رمز شده آن بوسیله کلید عمومی کاربر ساخته می شود

```
self.KEY, EnSharedKey = self.getSharedKey()
```

تولید عدد nonce هم بوسیله تابع (gen_nonce) انجام می شود .

و دو تابع مهم زیر وظیفه رمز کردن و باز کردن رمز و چک کردن integrity را برعهده دارند :

```
def readSocketAndOutput(self, s) :
    global byeFlag
    while True :
        if byeFlag :
            # try :
            INandWrite1 = s.recv(20000)
            hashed, str1 = pickle.loads(INandWrite1)
            print("Encrypted message : " + str(str1))
            str1 = AES_.decrypt(self.KEYNonce.encode(), str1)
            print("Delivered hash " + str(hashed))
            if hashed == hasher(str1) :
                logging.log("We Have integrity")
            else :
                logging.log("We Dont Have integrity")
            print("\r>>> " + str1 + "\n<<<", end = "", flush = True)
```

```
def readSTDINandWriteSocket(self, s) :
    global byeFlag
    while True :
        if byeFlag :
            str2 = input("<<< ")
            INandWrite2 = (hasher(str2), AES_.encrypt(self.KEYNonce.encode(), str2))
            INandWrite2 = pickle.dumps(INandWrite2)
            s.send(INandWrite2)
```

سوال ۲)

الف) هر دو در دسته Virtual private network ها قرار دارند و شباهت هر دو در عمل رمزنگاری آن ها می باشد

Ipsec در لایه ی network فعالیت دارد در حالی که SSL در بین دو لایه Application و Transport فعالیت می کند . بنابراین SSL بسته های لایه Application را گرفته رمز کرده و سپس تحویل Transport می دهد و تغییری در لایه Transport ایجاد نمی شود حال آن که Ipsec در لایه network عملی مشابه را انجام می دهد .

Ipsec ، EndSystem ها را به یک دیگر متصل می کند ولی SSL برنامه های لایه کاربرد را به یک دیگر متصل می کند و همچنین SSL بر روی لایه ی transport که reliable است پیاده سازی شده است و به خوبی وضعیت بین دو EndSystem و شماره پکت ها را دارد.

ب) از آنجا که Ipsec در لایه network پیاده شده است طیف گسترده تری را برای پیاده سازی پوشش می دهد نظیر استفاده در روتر ها ولی SSL باید در EndSystem ها پیاده شود و همچنین استفاده از ۲ راه token and digital certificate برای احراز هویت از مزایای رقابتی دیگر است .

سوال ۳) همون موقع هم که KEY ها متفاوت بود و IV داشت خطرناک بود و اگه اتکر در موقعیت Chosen plain text قرار می گرفت می تونست بدلیل سرعت بالا تمام رشته های KEY را بدست بیاورد (خاصیت XOR : M را دارد C هم دارد ، IV ها پلین تکست هستند ، در نتیجه K هم دارد)

حال اگه KEY ها یکسان بشه دیگه بدتر از بدتر چون راحت تر میتونه KEY را بازیابی کنه و نیازی به مشکلات IV نیست.

حالا اگه KEY یه جا قبلا هم استفاده شده باشه به منظور حرکتی دیگه ، دیگه بیا و جمعش کن اگر کلیدی به منظور احراز هویت استفاده شده باشه علاوه بر این که اون فردی که احراز هویت شده می تونه به تمامی پیام ها دسترسی داشته باشه (چه برای اون باشه چه نباشه) امکان لو رفتن هم بیشتر میشه (طبق توضیحات)

سوال ۴)

action	Source address	Dest address	protocol	Source port	Dest Port	Flag bit	C C
allow	Internal IP	any	tcp	>۱۰۲۴	۸۰	any	
allow	External ip	any	tcp	۸۰	>۱۰۲۴	syn	
allow	External ip	any	tcp	۸۰	>۱۰۲۴	any	+
allow	External ip	۱۷۶,۱۰۱,۵۱,۱۱۰	tcp	any	۸۰	any	
allow	External ip	۱۷۶,۱۰۱,۵۱,۱۱۰	tcp	any	۴۴۳	any	
allow	any	۱۷۶,۱۰۱,۵۱,۱۲۳	tcp	any	۲۲	any	
deny	all	all	all	all	all	all	

مبدأ	پورت مبدأ	مقصد	پورت مقصد
یه ip داخلی ای	۴۴۴۴	به یه داخلی دیگه	۸۰
یه ip داخلی ای	۴۴۴۵	به یه خارجی	۸۰

اخه چی بگم توی جدول کانکشن !

راستی در مورد اون خط ۲ و ۳ : چون نیازه که من وقتی یه سایتی را resolve می کنم اونم یه هندشیک انجام بده نیازه که پکت هایی که syn دارند بتونند بیان تو مگر این که بشه جواری رول نوشت که اگه من ازش درخواست resolve کردم این بره توی لیست کانکشن ها (با وجود این که کانکشنی بسته نشده و صرفا درخواست شده) که وقتی جوابش میاد اوکی باشه

سوال ۵)

خیر - چون اتکر هیچ کلیدی ندارد و Mac درستی نمی تواند قرار دهد تا از گام احراز هویت رد شود.

سوال ۶)

الف) با توجه به قانون tit-for-tat خیر | چرا که swarm متوجه شود bob با او همکاری ندارد لیستش را آپدیت کرده و جایگزین می کند و سهم کمی از چانک را ممکن است با این کار بدست آورده باشد . ولی خب شاید تعداد زیادی این فایل را داشته باشند و اون وقت داشته باشه با هرکدوم این حرکت را بزنه

ب) اگر سیستم هایش با یکدیگر هماهنگ باشد و هرکدام چانک های متفاوتی را دریافت کنند و تعدادشان هم کافی باشد می تواند بخش زیادی از chunk ها را دانلود کند.

سوال ۷) اگر فقط هرکسی از قبلی و بعدی اش خبر داشته باشد .

نود ۳ به نود ۴ ، نود ۴ به نود ۵ ، نود ۵ به نود ۸ اطلاع می دهند و چون نود ۸ می داند که نود بعدی آن می تواند نود ۹ باشد و همچنین نود بعدی خود را می داند که ۱۰ است .

نود ۱۰ به نود ۹ میدهد که خود را به او بشناساند و نود بعدی خود قرار دهد و P انود ۸ ، نود ۸ هم نود بعدی خود را آپدیت می کند و به این صورت نود ۹ وارد سیستم می شود .

