



## پروژه درس امنیت داده و شبکه

### اعضای گروه:

امیررضا میرزایی ۹۸۱۰۶۱۱۲

محمد آرمان سلیمانی ۹۸۱۰۵۸۳۵

محمدعلی خدابنده‌لو ۹۸۱۰۱۴۸۲

### فهرست

2.....	نحوه‌ی اجرای پروژه
2.....	توضیح کلی نحوه‌ی ارتباط
3.....	نیازمندی‌های امنیتی
4.....	فعالیت امتیازی
5.....	سناریو تیست

## نحوه اجرای پروژه

برای اجرای پروژه ابتدا پکیج های داخل requirement.txt را نصب کنید. سپس فایل server.py را در فolder سرور اجرا کنید تا سرور را بالا بیاورید. سپس فایل client.py در فolder client را اجرا کنید تا یک کلاینت بالا بیاورید. در صورتی که کلاینت های بیشتری می خواهید باز هم فایل client.py را به تعداد کلاینت ها اجرا کنید. سپس از طریق منوی کلاینت می توانید از پیام رسان امن استفاده کنید.

## توضیح کلی نحوه ارتباط

**ارتباط بین سرور و کلاینت :** پس از وصل شدن سوکت بین کلاینت و سرور، کلاینت یک کلید متقاضان برای رمز کردن ارتباط بین خودش و سرور تولید می کند و آن را با کلید عمومی سرور رمز می کند و برای سرور ارسال می کند. به این کلید client\_server\_sym\_key می گوییم. پس از اولین پیغام تمامی پیغام های رد و بدل شده بین کلاینت و سرور از طریق رمزگذاری و رمزگشایی با این کلید انجام می شود. همچنین این کلید به طور خود به خود پس از اینکه به تعداد مشخصی پیغام بین کلاینت و سرور رو بدل شد دوباره ایجاد و تبادل می شود. برای تبادل آن دوباره از کلید عمومی سرور استفاده می کنیم تا اگر مهاجم کلید فعلی را می داند نتواند کلید جدید را پیدا کند.

**رمزگذاری انتها به انتها بین کلاینت ها:** هنگامی که یک کلاینت قصد دارد با کلاینتی که با آن کلید مشترک ندارد پیغام بفرستد ابتدا از سرور درخواست می کند تا فرایند تبادل کلید را شروع کند. برای این کار از الگوریتم دیفی هلمن استفاده می کنیم. تبادل کلید طی یک فرایند ۵ مرحله ای به شکل زیر انجام می شود. فرض کنید clientA می خواهد به clientB پیغام بدهد.

۱. clientA sends a key exchange request to server.
۲. server generate diffie-hellman public parameter  $a, q$ .
۳. server send  $a$  and  $q$  to clientA and clientB
۴. clientA and clientB both generate their secret value and generate  $y_A, y_B$  and send them to the server
۵. server forwards  $y_A$  to B and  $y_B$  to A.

پس از این فرایند هر دو کلاینت یک رمز مشترک دارند که از طریق کانال سرور رد و بدل شده است اما سرور از آن اطلاع ندارد. به این کلید client\_client\_sym\_key می گوییم.

**ارسال پیغام بین ۲ کلاینت:** پس از تبادل کلید کلاینت پیغام خود را با `client_client_sym_key` رمز می‌کند و سپس به سرور می‌فرستد (پس از رمزگذاری با `client_server_sym_key`). سرور پیغام را رمزگشایی می‌کند و با کلید مشترک خودش با کلاینت دوم رمز می‌کند و برای کلاینت دوم ارسال می‌کند.

دقت کنید محتوای که سرور می‌بیند رمزشده‌ی پیغامی است که کلاینت اول فرستاده است.

**منطق گروه:** منطق گروه به شکل بسیار ساده اما امنی پیاده شده است. ابتدا دقت کنید زمانی که ادمین فردی را به گروه اضافه می‌کند با آن فرد یک کلید متقارن با منطقی که توضیح دادیم می‌کند. پس ادمین با تمام اعضای گروه یک کلید متقارن دارد.

حال فرض کنید عضوی از گروه می‌خواهد برای دیگر اعضا پیغام ارسال می‌کند ۲ حالت داریم:

۱. اگر فرد ادمین گروه باشد:

در این صورت این فرد با تمام اعضای گروه یک کلید مشترک دارد پس پیغام را به تمام آن‌ها می‌فرستد.

۲. اگر فرد ادمین گروه نباشد:

در این صورت ممکن است این فرد با تمام اعضای گروه کلید مشترک نداشته باشد پس پیغام را برای ادمین ارسال می‌کند و ادمین

مانند حالت ۱ پیغام را برای دیگر اعضا می‌فرستد.

دقت کنید این روش کارا نیست زیرا لود زیادی سمت ادمین است اما امنیت کامل دارد.

### نحوه‌ی نگهداری کلید و پیغام‌ها در سمت کلاینت:

از هر کلاینت هنگام لاگین یک پسورد می‌گیریم و از طریق این پسورد یک کلید متقارن ایجاد می‌کنیم. سپس هم کلیدها و هم پیغام‌ها را با این کلید به صورت رمز شده نگه می‌داریم و کاربر برای دسترسی به آن‌ها باید پسورد خود را وارد کند. دقت کنید پسورد اولیه در برنامه به صورت یک متغیر وجود دارد اما کاربر می‌تواند آن را عوض کند و در صورت عوض کردن آن دیگر این پسورد را نگه نمی‌داریم.

دقت کنید دلیل نگه داشتن پسورد اولیه این است که کلاینت هنگامی که پیغام دریافت می‌کند و یا طرف دوم تبادل کلید است (طرفی که تبادل را شروع نکرده است) باید رمز و یا پیغام را با چیزی رمز کنیم به همین منظور از یک پسورد اولیه استفاده می‌کنیم.

### نیازمندی‌های امنیتی

**۱- رمزگذاری انتهای همانطور که توضیح داده شد تمامی پیغام‌های بین ۲ کلاینت با کلید مشترک آن‌ها رمز شده است و سرور فقط آن را فوروارد می‌کند و نمی‌تواند محتویات آن را بخواند.**

**۲- تازگی کلید:** فرایند تبادل کلید بین ۲ کلاینت از طریق الگوریتم دیفی‌هلمن انجام می‌شود و پارامترهای عمومی آن را سرور اعلام می‌کند پس یک کلاینت نمی‌تواند کلید قبلی خود را به عنوان کلید جلسه اعلام کند.

**۳- محرومگی:** تمامی پیغام‌های بین کلاینت و سرور از طریق یک کلید مشترک رمز شده است همچنین پیغام‌های بین ۲ کلاینت از طریق رمزگذاری انتهای همانطور که توضیح داده شده است.

**۴- صحبت و یکپارچگی:** هم سرور و هم کلاینت قبل از رمزگذاری پیامی که می‌خواهند ارسال کنند در آن یک هش می‌گیرند و در کنار پیغام اصلی رمز می‌کنند. همچنین هر دو هنگام دریافت کردن یک پیغام درستی هش دریافتی با هش خود پیام را چک می‌کنند.

**۵- حفظ سازگاری:** سرور فقط یک سوکت با هر کلاینت دارد پس یک کلاینت نمی‌تواند در یک لحظه چند پیغام ارسال کند. همچنین هر کلاینت پس از اینکه پیغام خود را ارسال می‌کند منتظر تایید سرور می‌شود. با این کار مطمئن می‌شویم که پیغام‌ها به ترتیب به کلاینت دیگر می‌رسند. پس کافیست در کلاینت هم پیغام‌های دریافتی را به ترتیب نگه داریم.

**۶- احراز اصالت:** پس از اینکه کلاینت لاگین می‌کند کلید مشترک بین این کلاینت و سرور به همراه یوزرنیم کلاینت ذخیره می‌شود. در مراحل بعد اگر کلاینت ادعا کند یک یوزر مشخص است سرور کلید پیغام دریافتی را با کلید ذخیره شده هنگام لاگین مقایسه می‌کند و از این طریق مطمئن می‌شود که این کلاینت همانی است که قبلاً با این یوزرنیم لاگین کرده است.

**۷- عدم انکار:** سرور روی پیغام‌های ردوبدل شده بین ۲ کلاینت امضا می‌زند. بدین شکل که پیام ارسالی از کلاینت اول را هش می‌کند و سپس با کلید خصوصی خود رمز می‌کند و هنگام فوروارد این امضا را در کنار پیغام به کلاینت دوم می‌رساند.

**۸- کنترل دسترسی:** سرور پس از دریافت درخواست اضافه یا حذف کردن کاربر از گروه سطح دسترسی فرستنده را چک می‌کند و در صورت نداشتن دسترسی پیغام مناسب را برای فرستنده درخواست می‌فرستد. دقت کنید سرور اطلاعات گروه‌ها را دارد.

**۹- حمله‌ی مردی در میان:** تمامی پیغام‌های بین کلاینت و سرور با یک کلید متقاضن رمز شده‌اند و مهاجم نمی‌تواند محتویات پیغام‌ها را بخواند.

**۱۰- حمله‌ی تکرار:** مقاله‌ی با این حمله مانند مکانیزم پیاده شده در **ipsec timestamp** است بدین شکل که هم کلاینت و هم سرور در هنگام ارسال پیغام یک پیغام اضافه می‌کنند همچنین یک عدد بین ۰ تا ۹ نیز به پایان پیغام اضافه می‌کند. این عدد هر بار یکی اضافه می‌شود.

همچنین کلاینت و سرور یک آرایه به طول ۱۰ دارند که هنگام دریافت یک پیغام **timestamp** و عدد را استخراج می‌کند. سپس در آرایه به خانه‌ی آن عدد می‌رود. در اینجا ۲ حالت داریم:

- خانه‌ی آرایه خالی است در این صورت **timestamp** را در آن می‌نویسیم.

- خانه‌ی آرایه خالی نیست در صورتی که عدد داخل آن قبل از **timestamp** فعلی باشد **timestamp** فعلی را در آن می‌نویسیم، اما اگر اینطور نباشد حمله‌ی تکرار رخ داده است.

همچنین دقت کنید که خود **timestamp** و شماره‌ی خانه‌ی آرایه نیز با کلید بین کلاینت و سرور رمز شده‌اند.

**۱۱- استفاده از الگوریتم‌های رمزنگاری امن:** برای کلید عمومی و خصوصی سرور از RSA با کلید ۲۰۴۸ بیتی استفاده می‌کنیم. برای امضا از sha۲۵۶ و RSA استفاده می‌کنیم. دقت کنید از RSA در حالت امن آن برای امضا استفاده می‌کنیم. همچنین برای رمز متقاضن از الگوریتم fernet در کتابخانه cryptography پایتون استفاده می‌کنیم که بر پایه AES است اما در مقابل حملاتی مثل افزایش طول مقاوم است.

**۱۲- محروم‌نگی پیشرو و محروم‌نگی آینده:** بدلیل استفاده از دیفری هلمن در تبادل کلید هر دوی این ویژگی‌ها برقرار است.

## فعالیت امتیازی

### ۱- حذف کاربر از گروه:

بدلیل نحوه‌ی پیاده‌سازی گروه کافیست در سمت سرور کاربر را از دیتابیس گروه حذف کنیم.

### ۲- ارسال پیام به کاربر آفلاین:

این مورد پیاده نشده است اما نحوه‌ی پیاده سازی آن طراحی شده است بدین شکل که هر کاربر پس از لگین یک پارامتر  $a$  و  $q$  و  $y$  به سرور می‌فرستد. از این ۳ عدد برای تولید کلید از طریق الگوریتم دیفری هلمن به صورت افلاین استفاده می‌شود. در صورتی که کاربر دیگری بخواهد به کاربر افلاین پیغام بدهد سرور ۳ پارامتر را برایش می‌فرستند و کلاینت با تولید یک کلید خصوصی کلید مشترک را تولید می‌کند.

سپس کاربر انلاین  $u$  مربوط به خود را به سرور می‌فرستد تا ذخیره شود. حال هر پیغامی که کاربر انلاین بخواهد به کاربر افلاین بفرستند را با کلید مشترک رمز می‌کنیم و به سرور می‌فرستیم. سرور تمام این پیغام‌ها را ذخیره می‌کند و هنگامی که کاربر افلاین لگین می‌کند سرور تمام پیغام‌های ذخیره شده را به کاربر می‌فرستند. کاربر می‌تواند با  $y$  دریافتی از کلاینت و  $X$  خود به کلید مشترک برسد و پیغام‌ها را رمزگشایی کند.

سناریو تست

در این ستاریو ۳ کلاینت های بالای صفحه کلاینت هستند و صفحه‌ی پایین ترمینال سرور است. در ابتدا پس از وصل شدن ۳ کلاینت یک کلید متقارن از طریق کلید عمومی سرور تبادل می‌شود. (می‌توانید پیغام‌ها را در صفحه‌ی سرور بینند).

سپس از چپ به راست ۳ کاربر با نام‌های amir ، reza ، arman می‌سازیم. همانطور که می‌بینید پسورد کلاینت‌ها به صورت هش شده ارسال می‌شود.

The screenshot displays three separate macOS Terminal windows, each showing the execution of a Python script named `client.py`. The script interacts with a service running on port 8024, as indicated by the command `curl -X POST http://127.0.0.1:8024`.

**Terminal 1:**

```
curl -X POST http://127.0.0.1:8024
1> enter your desired username:amir
please enter a password:1234
user amir successfully registered.
1> create account
2> login
3> logout
4> show online users
5> send message
6> message history
7> set keychain password
8> check session integrity
9> refresh session key
10> get groups info
11> add user to group
12> remove user from group
13> send message to group
```

**Terminal 2:**

```
curl -X POST http://127.0.0.1:8024
1> enter your desired username:reza
please enter a password:1234
user reza successfully registered.
1> create account
2> login
3> logout
4> show online users
5> send message
6> message history
7> set keychain password
8> check session integrity
9> refresh session key
10> get groups info
11> add user to group
12> remove user from group
13> send message to group
```

**Terminal 3:**

```
curl -X POST http://127.0.0.1:8024
1> enter your desired username:arman
please enter a password:1234
user arman successfully registered.
1> create account
2> login
3> logout
4> show online users
5> send message
6> message history
7> set keychain password
8> check session integrity
9> refresh session key
10> get groups info
11> add user to group
12> remove user from group
13> send message to group
```

The output shows the user interacting with the service through various commands, such as creating accounts, logging in, sending messages, and checking session integrity.

سپس از طرف هر ۳ کلاینت لاگین می‌کنیم. پس از لاغین از هر کاربر یک پسورد برای رمزنگاری کلیدها و پیغام‌های ایش می‌گیریم.

سپس از طرف amir به reza پیغام می‌فرستیم. همانطور که می‌بینید در ابتدا فرایند تبادل کلید انجام شده است. همچنین reza و arman تا قبل از زدن رمز عبور نمی‌توانند رمزگشایی شده‌ی پیغام دریافتی را ببینند. همچنین در تصویر می‌توانید امضا پیغام‌ها را در ترمینال سرور ببینید.

سپس هر کاربر می‌تواند لیست تمام پیغام‌های دریافتی خود را با زدن پسورد کلید خود بزنند.

```
client — IPython: DNS-Project/client — Python client.py — 80x24
enter key chain password:
amir$amirk
enter message list old password:
enter new password for your message list:
amir$amirk
sent to rezz: Hi reza, Im amir
from amir: Hi amir, Im amirk
sent to amirk: Hi amirk, Im amirk
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-get session key
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amirk|
```

```
client — Python client.py — 80x24
6
enter key chain password:
amir$amirk
enter message list old password:
rezz$rezz
enter new password for your message list:
amirk$amirk
received from amir: Hi reza, Im amir
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-get session key
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amirk|
```

```
client — Python client.py — 80x24
6
enter key chain password:
amir$amirk
enter message list old password:
enter new password for your message list:
amirk$amirk
received from amir: Hi amirk, Im amir
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-get session key
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amirk|
```

همچنین می‌توانیم صحت نشست را با تبدیل کلید به اموجی بررسی کنیم.

همچنین هر کاربر قابلیت بازسازی کلید نشست خود با کلاینت‌های دیگر را نیز دارد برای مثال در تصویر زیر کلید بین amir و reza را دوباره ساختیم.

```
Terminal Shell Edit View Window Help
client - Python client.py - 80x24
enter key chain password:
amir/amirk
reza@: ~ %
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amir@ changing key with reza
changing key with reza complete.

reza@: ~ %
14-send message to group

reza@: ~ %
enter username of person you wish to exchange key with again:
multiple choices: amir
reza@: ~ %
reza@: ~ %
exchanging key with amir
exchanging key with amir complete.

reza@: ~ %
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

reza@: ~ %
amir@ client - Python client.py - 80x24
14-add user to group
15-remove user from group
14-send message to group

amir@: ~ %
enter key chain password:
amir/amirk
14-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amir@: ~ %
```

حال یک گروه با ادمینی amir و اعضای reza و arman می‌سازیم. سپس در ترمینال reza و arman اطلاعات گروهها را از سرور می‌گیریم.

```
client — IPython: DNS-Project/client — Python client.py — 80x24
amir[2]
enter the username of the person you want to add to group:
enter the group you want to add aman to:
group added to group
initializing key exchange...
Key already set.
create account
2-login
3-logout
show online users
6-send message
7-set keychain password
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amir[3]
Run: !server
received after decryption: {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman'}
received {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman'}
sending: b'{"api": "exchange_p1", "username": "aman", "g": 643344081958547189807587381577622612721301236121715434008425651557312194969991155625947248024914206085721286554651605956738245138245382025809764923, "g": 2}'
sending: b'{"api": "exchange1", "username": "reza", "g": 84334408195854571898075873615772271301236121715434008425651557312194969991155625947248024914206085721286554651605956738245138245382025809764923, "g": 2}'

received before decryption: b'gAAAAABuQh0IVu9mF0D709y39cMu1nbKwLQfj1A75Lslen1KgHs3nKM8esinBvmoTxBmpLwXjwJ1RJEWpTICB0gIaD6pG-101-1h2-JtWypG0aB1piqEcNaen9h0V7t6QFQcQ0 -gt6E0K3CKHxKC-Ndm#W9hLx3Z562-ePmtW0oCjB6GZCt-0ZMw_nCP50XusU151S0_uLo5xR1k0B4pP99KKe0z27505p0u7rX04eCeDy97F2H06xGzT52a0nU0tUdDdc77xLcfz-7xvIobjTyuWnDnyy_C0Nhwx4_0hIy5X21Drbz -977g1TkSz219yJyoxognPb7469_r3m0YBz_P-F9MKtBhAgltwtcrp12osg-FcbnXKm0DzQw!VbJh32inNl08wIc14M7P-rUsJgJzFdNgxN9J_-jeQwAD1'
received after decryption: {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman', 'g': 20945653247126739308356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}
received {'api': 'exchange_with_another_client_p2', 'sender': 'reza', 'receiver': 'aman', 'g': 209456532471266793080356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}
sending: b'{"api": "exchange_p2", "username": "reza", "g": 209456532471266793080356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}'
```

```
client — Python client.py — 80x24
16-add group
17-get groups info
18-add user to group
19-remove user from group
20-send message to group

reza[1]
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

reza[2]
16-add group
17-get groups info
18-add user to group
19-remove user from group
20-send message to group

amir[1]
1-create account
2-login
3-logout
4-show online users
5-send message
6-message history
7-set keychain password
8-check session integrity
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

arman[1]
```

```
client — IPython: DNS-Project/client — Python client.py — 80x24
amir[2]
enter the username of the person you want to add to group:
enter the group you want to add aman to:
group added to group
initializing key exchange...
Key already set.
create account
2-login
3-logout
show online users
6-send message
7-set keychain password
9-refresh session key
10-add group
11-get groups info
12-add user to group
13-remove user from group
14-send message to group

amir[3]
Run: !server
received after decryption: {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman'}
received {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman'}
sending: b'{"api": "exchange_p1", "username": "aman", "g": 643344081958547189807587381577622612721301236121715434008425651557312194969991155625947248024914206085721286554651605956738245138245382025809764923, "g": 2}'
sending: b'{"api": "exchange1", "username": "reza", "g": 84334408195854571898075873615772271301236121715434008425651557312194969991155625947248024914206085721286554651605956738245138245382025809764923, "g": 2}'

received before decryption: b'gAAAAABuQh0IVu9mF0D709y39cMu1nbKwLQfj1A75Lslen1KgHs3nKM8esinBvmoTxBmpLwXjwJ1RJEWpTICB0gIaD6pG-101-1h2-JtWypG0aB1piqEcNaen9h0V7t6QFQcQ0 -gt6E0K3CKHxKC-Ndm#W9hLx3Z562-ePmtW0oCjB6GZCt-0ZMw_nCP50XusU151S0_uLo5xR1k0B4pP99KKe0z27505p0u7rX04eCeDy97F2H06xGzT52a0nU0tUdDdc77xLcfz-7xvIobjTyuWnDnyy_C0Nhwx4_0hIy5X21Drbz -977g1TkSz219yJyoxognPb7469_r3m0YBz_P-F9MKtBhAgltwtcrp12osg-FcbnXKm0DzQw!VbJh32inNl08wIc14M7P-rUsJgJzFdNgxN9J_-jeQwAD1'
received after decryption: {'api': 'key_exchange_with_another_client_p1', 'sender': 'reza', 'receiver': 'aman', 'g': 20945653247126739308356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}
received {'api': 'exchange_with_another_client_p2', 'sender': 'reza', 'receiver': 'aman', 'g': 209456532471266793080356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}
sending: b'{"api": "exchange_p2", "username": "reza", "g": 209456532471266793080356491993330157515509837440112136321764097301209735209738603716166897807310128640684139374374187409335731985508940973052130546362}'
```

سپس amir یک پیغام به داخل گروه می‌فرستد و همانطور که می‌بینید ابتدا پیغام به amir به عنوان admin ارسال شده است و سپس amir آن را برای دیگر اعضا فرستاده است.