22162005 Nguyễn Lưu Gia Bảo

22162048 Bùi Lê Thuỷ Tiên

1. Giới thiệu về SSH public-key based Authentication và SCP

1.1 SSH public-key based Authentication

Trước khi xuất hiện SSH thì Telnet là giao thức chủ yếu được sử dụng để giao tiếp với một máy chủ từ xa. Tuy nhiên, Telnet không phải là một giao thức liên lạc an toàn vì nó không sử dụng bất kỳ cơ chế bảo mật nào và truyền dữ liệu qua mạng/Internet dưới dạng văn bản thuần túy, bao gồm cả mật khẩu. Do đó, bất kỳ ai cũng có thể nghe lén các gói tin để lấy thông tin quan trọng này. Để khắc phục vấn đề này, giao thức SSH đã ra đời.

SSH, còn được gọi là Secure Shell hoặc Secure Socket Shell, là một giao thức mạng cung cấp cho quản trị viên một cách an toàn để truy cập vào máy tính từ xa. SSH thiết lập một kết nối được mã hóa giữa hai bên (máy khách và máy chủ), xác thực từng bên với bên còn lại, đồng thời truyền lệnh và kết quả qua lại giữa hai bên.

Giao thức SSH sử dụng mã hóa đối xứng, mã hóa bất đối xứng và băm (hashing) để bảo vệ quá trình truyền thông tin. Quá trình thiết lập kết nối SSH giữa máy khách và máy chủ diễn ra trong ba giai đoạn:

Xác minh máy chủ bởi máy khách.

- Máy khách khởi tạo kết nối SSH với máy chủ. Máy chủ lắng nghe các kết nối SSH trên cổng mặc định 22 (có thể thay đổi). Tại thời điểm này, danh tính của máy chủ sẽ được xác minh. Có hai trường hợp:

- Nếu máy khách truy cập máy chủ lần đầu tiên, máy khách được yêu cầu xác thực máy chủ theo cách thủ công bằng cách xác minh khóa công khai của máy chủ. Khóa công khai của máy chủ có thể được tìm thấy bằng lệnh ssh-keyscan hoặc có thể được tìm thấy ở những nơi khác nhau. Sau khi khóa được xác minh, máy chủ được thêm vào tệp known_hosts trong thư mục ~/.ssh trên máy khách. Tệp known_hosts chứa thông tin về tất cả các máy chủ đã được máy khách xác minh.

- Nếu máy khách không truy cập máy chủ lần đầu tiên, danh tính của máy chủ sẽ được so khớp với thông tin đã ghi lại trước đó trong tệp known_hosts để xác minh.

Tạo khóa phiên (session key) để mã hóa toàn bộ quá trình giao tiếp.

Sau khi máy chủ được xác minh, cả hai bên sẽ thương lượng khóa phiên bằng một phiên bản của cái gọi là thuật toán Diffie-Hellman. Thuật toán này được thiết kế theo cách mà cả hai bên đều đóng góp như nhau trong việc tạo khóa phiên. Khóa phiên được tạo ra là khóa đối xứng được chia sẻ (shared symmetric key), tức là cùng một khóa được sử dụng để mã hóa và giải mã.

Xác thực máy khách.

Giai đoạn cuối cùng liên quan đến việc xác thực máy khách. Việc xác thực được thực hiện bằng cặp khóa SSH. Như tên gọi, cặp khóa SSH bao gồm hai khóa phục vụ hai mục đích khác nhau:

- Khóa công khai (public key) được sử dụng để mã hóa dữ liệu và có thể chia sẻ tự do.

- Khóa riêng tư (private key) được sử dụng để giải mã dữ liệu và không bao giờ được chia sẻ với bất kỳ ai.

Sau khi mã hóa đối xứng đã được thiết lập, quá trình xác thực máy khách diễn ra như sau:

Máy khách bắt đầu bằng cách gửi ID của cặp khóa mà nó muốn sử dụng để xác thực đến máy chủ.

Máy chủ kiểm tra tệp authorized_keys của tài khoản mà máy khách đang cố gắng đăng nhập để tìm ID khóa.

Nếu tìm thấy khóa công khai có ID khớp trong tệp, máy chủ sẽ tạo một số ngẫu nhiên, sau đó sử dụng khóa công khai để mã hóa số này và gửi thông điệp đã mã hóa cho máy khách.

Nếu máy khách có khóa riêng tư chính xác, nó sẽ giải mã thông điệp để lấy số ngẫu nhiên do máy chủ tạo ra.

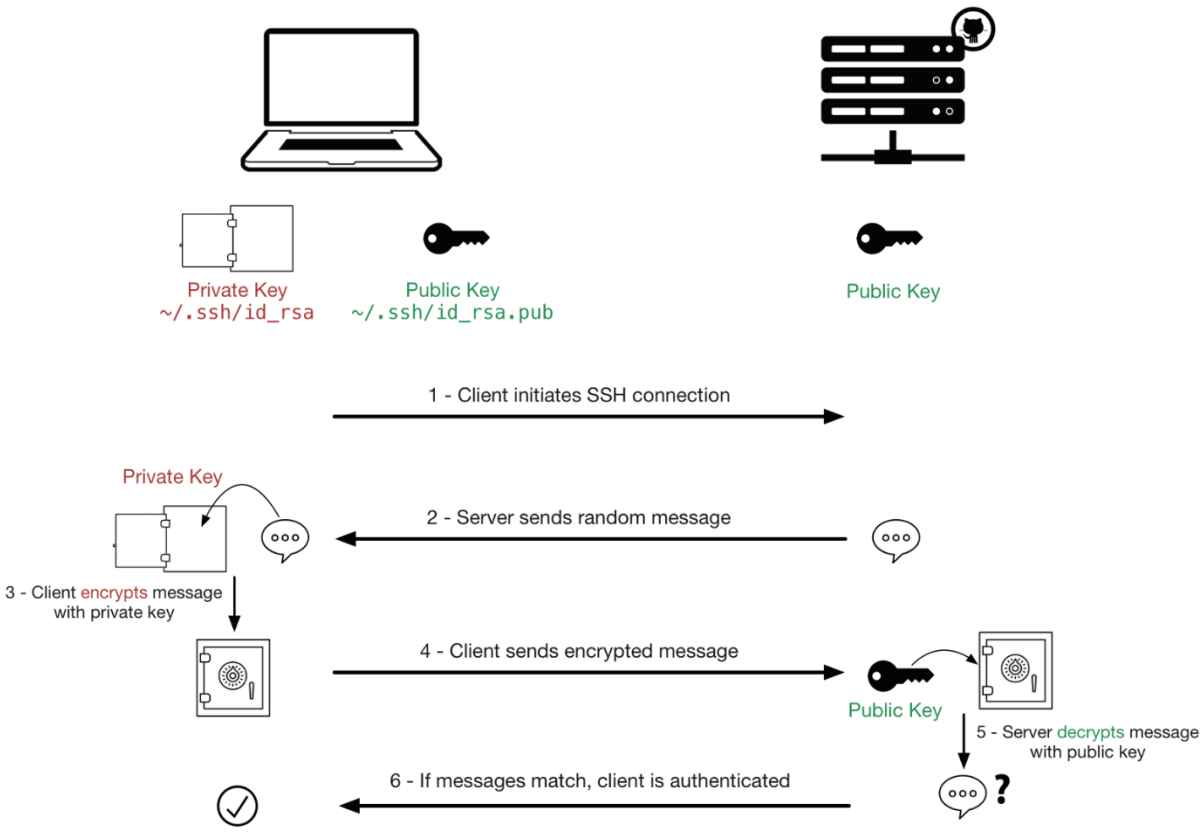
Máy khách kết hợp số ngẫu nhiên nhận được với khóa phiên chung và tính toán giá trị băm MD5 của kết quả này.

Máy khách gửi giá trị MD5 hash này trở lại cho máy chủ như một phản hồi cho thông điệp đã mã hóa trước đó.

Máy chủ sử dụng cùng khóa phiên chung và số ngẫu nhiên gốc mà nó đã gửi cho máy khách để tự tính toán giá trị MD5. Sau đó, nó so sánh kết quả của mình với giá trị MD5 mà máy khách đã gửi lại.

Nếu hai giá trị trùng khớp, điều đó chứng tỏ máy khách đang sở hữu khóa riêng tư hợp lệ và quá trình xác thực hoàn tất thành công.

Tính bất đối xứng của cặp khóa cho phép xác thực máy khách, vì máy khách chỉ có thể giải mã thông điệp nếu nó sở hữu khóa riêng tư chính xác tương ứng với khóa công khai của máy chủ.



1.2. SCP (Secure Copy Protocol)

SSH dùng để truyền tệp và thư mục giữa hai máy tính. SCP cho phép sao chép tệp và thư mục từ hệ thống cục bộ sang hệ thống từ xa hoặc ngược lại. Nó dựa trên giao thức SSH, vì vậy ta phải cung cấp mật khẩu SSH để xác thực hệ thống từ xa trước khi sao chép tệp. Với SCP có thể copy file hoặc thư mục:

- Copy file, thư mục từ máy local lên server

- Copy file, thư mục từ máy server vè client

- Copy file, thư mục từ máy server này sang máy server khác.

Khi truyền dữ liệu bằng scp, cả tệp và mật khẩu đều được mã hóa để bất kỳ ai theo dõi lưu lượng truy cập đều không nhận được bất kỳ thông tin nhạy cảm nào.

2. Phân công thực hiện nhiệm vụ của từng thành viên.

Các chương trình này bao gồm các phần sau:

scp_key_client.c: Kết nối tới máy tính từ xa sử dụng SSH, xác thực bằng Public-key, và sao chép các file từ thư mục source_dir trên máy cục bộ tới máy tính từ xa.

ssh_server.c: Lắng nghe các kết nối SSH, xác thực bằng Public-key, và nhận các file được sao chép từ máy cục bộ.

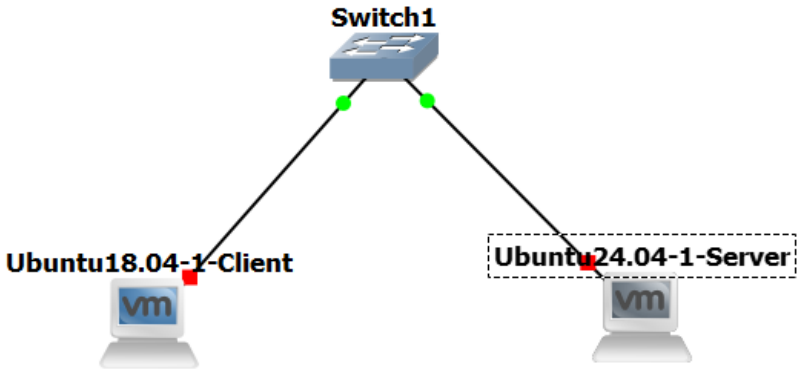
Phân công nhiệm vụ cho từng thành viên như sau:

Thành viên 1 (22162005): Viết chương trình scp_key_client.c

Thành viên 2 (22162048): Viết chương trình ssh_server.c

3. Thiết lập môi trường thực thi

Máy client và server kết nối cùng mạng với nhau và thông qua switch như mô hình mạng đơn giản như ảnh sau




```
nhoclahola@ubuntu:~/Documents/kiem_Tra$ sudo apt install libssh-4 libssh-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
libssh-4 is already the newest version (0.6.3-4.3ubuntu0.6).
The following packages were automatically installed and are no longer required:
  bridge-utils cgroupfs-mount containerd default-jdk-headless docker.io
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libx11-doc libxau-dev
  libxcb1-dev libxdmcp-dev libxt-dev openjdk-8-jdk openjdk-8-jdk-headless pigz
  runc snapd-login-service ubuntu-fan x11proto-core-dev x11proto-input-dev
  x11proto-kb-dev xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
Suggested packages:
  libssh-doc
The following NEW packages will be installed:
  libssh-dev
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 169 kB of archives.
After this operation, 719 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

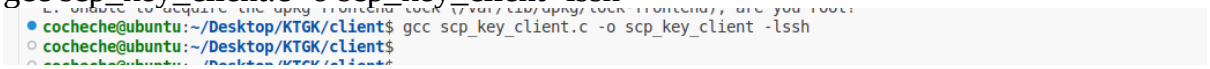
*** Thực thi và biên dịch chương trình scp_key_client.c**

- Cài đặt thư viện libssh-dev

sudo apt update && apt install libssh-dev

- Biên dịch chương trình scp_key_client.c

gcc scp_key_client.c -o scp_key_client -lssh

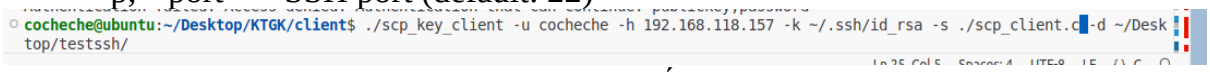


- Thực thi chương trình scp_key_client

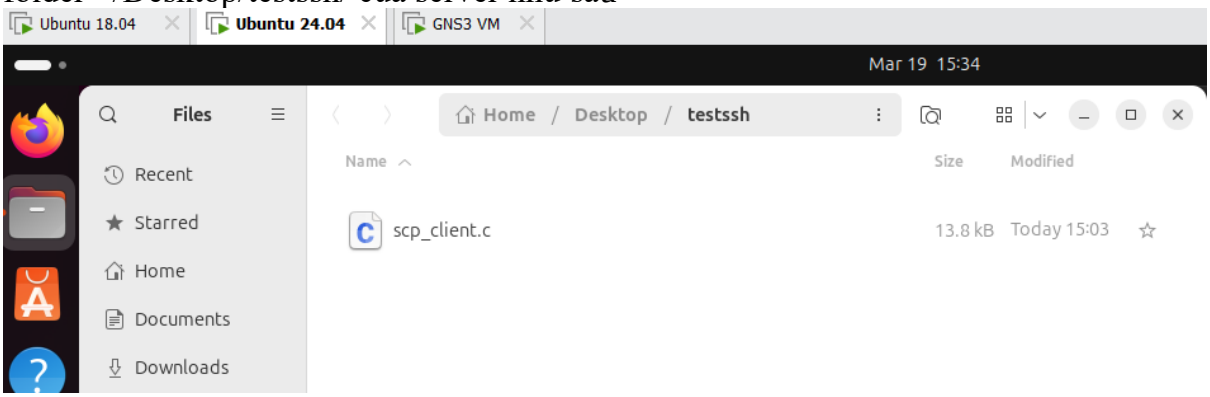
./scp_key_client -u cocheche -h 192.168.118.157 -k ~/.ssh/id_rsa -s ./scp_client.c -d ~/Desktop/testssh/

Với các tham số như sau:

-u, --user	SSH username
-h, --host	SSH host/IP address
-k, --key	Path to private key file
-s, --source	Local file or directory to copy
-d, --dest	Remote destination path
-p, --port	SSH port (default: 22)



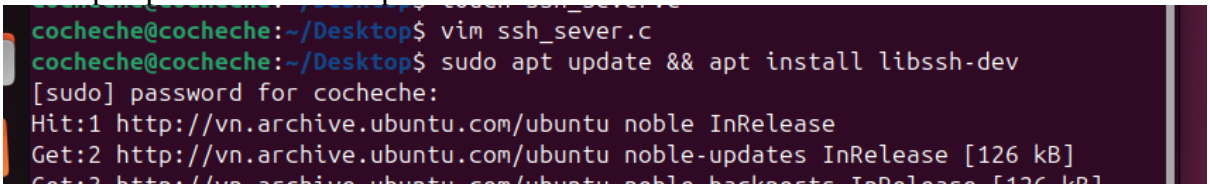
Sau khi thực thi lệnh thì lệnh này thì client sẽ tiến hành gửi file scp_client.c sang folder ~/Desktop/testssh/ của server như sau



*** Thực thi và biên dịch chương trình ssh_server.c**

- Cài đặt thư viện libssh-dev

sudo apt update && sudo apt install libssh-dev



- Biên dịch chương trình ssh_server.c

gcc ssh_sever.c -o ssh_server -lssh

```
Setting up libssh-dev:amd64 (0.10.6-2build2) ...
Processing triggers for man-db (2.12.0-4build2) ...
cocheche@cocheche:~/Desktop$ gcc ssh_sever.c -o ssh_server -lssh
cocheche@cocheche:~/Desktop$
```

- *Thực thi chương trình ssh_server*

```
cocheche@cocheche:~/Desktop$ ^C
cocheche@cocheche:~/Desktop$ ./ssh_server
```

Sau khi chạy lệnh thì server sẽ lắng nghe ở port 2222 để nhận file từ scp client gửi về.

4. Giải pháp thực hiện (Cách tiếp cận)

Bài toán được đặt ra là sao chép an toàn (secure copy) nhiều file chứa trong 1 thư mục source trên máy cục bộ (local machine) đến 1 thư mục dest trên máy tính ở xa.

Để có thể truyền file giữa client và server, nhóm quyết định sẽ dựa vào openssh-server trước, và dựa vào đó để viết giao thức truyền tin SCP thông qua giao thức SSH mà server mẫu ở đây sẽ chạy openssh để có thể kiểm tra tính đúng đắn của của code của mình.

Đầu tiên là cài đặt openssh-server trên 1 máy làm SSH Server.

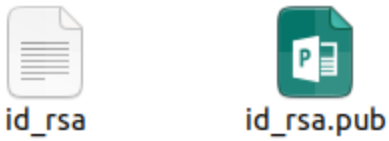
```
nhoclahola@ubuntu:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:7.2p2-4ubuntu2.10).
The following packages were automatically installed and are no longer required:
  bridge-utils cgroupfs-mount containerd default-jdk-headless docker.io
  libice-dev libpthread-stubs0-dev libsm-dev libx11-dev libx11-doc libxau-dev
  libxcb1-dev libxdmcp-dev libxt-dev openjdk-8-jdk openjdk-8-jdk-headless pigz
  runc snapd-login-service ubuntu-fan x11proto-core-dev x11proto-input-dev
  x11proto-kb-dev xorg-sgml-doctools xtrans-dev
Use 'sudo apt autoremove' to remove them.
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

Tại máy server này có một người dùng tên là "nhoclahola".

Sau đó tại máy làm client sẽ tạo ra các cặp khoá riêng tư và công khai cho người dùng này:

```
nhoclahola@nhoclahola-vm:~/Key$ ssh-keygen -t rsa -b 4096 -C "nhoclahola@gmail.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/home/nhoclahola/.ssh/id_rsa):
/home/nhoclahola/.ssh/id_rsa already exists.
Overwrite (y/n)?
```

Được 2 file là id_rsa và id_rsa.pub chứa khoá riêng tư và khoá công khai.



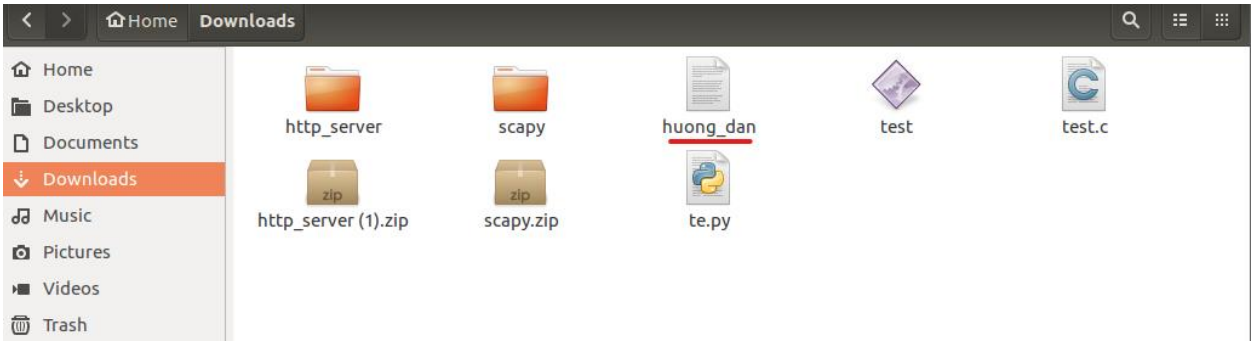
Sau đó copy khoá công khai để cho server xác thực.

```
nhoclahola@nhoclahola-vm:~/Key$ ssh-copy-id nhoclahola@192.168.119.132
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already install
ed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new
keys
nhoclahola@192.168.119.132's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'nhoclahola@192.168.119.132'"
and check to make sure that only the key(s) you wanted were added.
```

Cuối cùng là thử nghiệm truyền file scp thông qua key vừa tạo, ví dụ truyền một file tên huong_dan vào folder Downloads của người dùng SSH.

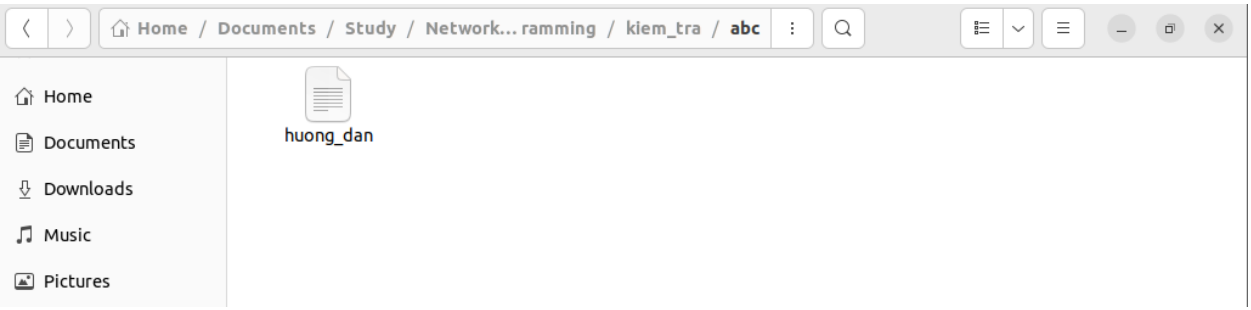
```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra$ sudo scp
 -i ~/.ssh/id_rsa huong_dan nhoclahola@192.168.119.132:~/Downloads
huong_dan                                     100%  107    82.9KB/s   00:00
```

Như vậy là truyền file thành công và file huong_dan sẽ xuất hiện trên folder Downloads của người dùng nhoclahola trên SSH server.



Hoặc ngược lại là tải copy file từ SSH server:

```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra/abc$ scp
-i ~/.ssh/id_rsa nhoclahola@192.168.119.132:~/Downloads/huong_dan ./
huong_dan                                     100%  107    81.7KB/s   00:00
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra/abc$
```



Dựa vào đó, nhóm trước tiên tiếp cận bằng cách làm SCP client trước tiên với một SSH server đã có sẵn, vì nó đã có sẵn SSH Server như vậy nên sẽ dễ dàng để xác minh là code của nhóm bị sai hay không.

5. Mô tả và thực thi

5.1. SCP Client

Đầu tiên sẽ viết code về truyền file (trên client) dựa trên ssh trước (để đảm bảo đúng đắn của server).

Chương trình sẽ chạy với các tham số như sau:

```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra$ ./scp_key_client
Error: Missing required arguments
Usage: ./scp_key_client -u <username> -h <host> -k <private_key_path> -s <source> -d <destination> [-p <port>] [
--download]
Options:
  -u, --user       SSH username
  -h, --host       SSH host/IP address
  -k, --key        Path to private key file
  -s, --source     Source path (local for upload, remote for download)
  -d, --dest       Destination path (remote for upload, local for download)
  -p, --port       SSH port (default: 22)
  --download       Download from server instead of upload
  --help           Display this help message

Example (upload): ./scp_key_client -u seed -h 172.19.0.2 -k ~/.ssh/id_rsa -s ./local_dir/ -d ~/remote_dir/
Example (download): ./scp_key_client -u seed -h 172.19.0.2 -k ~/.ssh/id_rsa -s ~/remote_dir/ -d ./local_dir/ --d
ownload
```

Cụ thể, workflow của nó sẽ như sau:

- Đầu tiên là nhận các tham số từ dòng lệnh.

```c
static struct option long_options[] = {
    {"user", required_argument, 0, 'u'},
    {"host", required_argument, 0, 'h'},
    {"key", required_argument, 0, 'k'},
    {"source", required_argument, 0, 's'},
    {"dest", required_argument, 0, 'd'},
    {"port", required_argument, 0, 'p'},
    {"download", no_argument, 0, 'D'},
    {"help", no_argument, 0, 'H'},
    {0, 0, 0, 0}
};

int opt, option_index = 0;
while ((opt = getopt_long(argc, argv, "u:h:k:s:d:p:DH", long_options, &option_index)) != -1) {
    switch (opt) {
        case 'u': username = optarg; break;
        case 'h': hostname = optarg; break;
        case 'k': priv_key_path = optarg; break;
        case 's': source_path = optarg; break;
        case 'd': dest_path = optarg; break;
        case 'p':
            port = atoi(optarg);
            if (port <= 0 || port > 65535) {
                fprintf(stderr, "Invalid port number: %s\n", optarg);
                exit(-1);
            }
            break;
        case 'D': download_mode = 1; break;
        case 'H': print_usage(argv[0]); exit(0);
        default: print_usage(argv[0]); exit(-1);
    }
}
```

- Sau đó xác tạo phiên SSH, thiết lập hostname, cổng, username và kết nối đến server, cũng như là tải khoá riêng tư được truyền từ tham số (thông qua đường dẫn file) dựa vào các hàm trong thư viện ssh.

```c
ssh_options_set(my_ssh_session, SSH_OPTIONS_USER, username);

int rc = ssh_connect(my_ssh_session);
if (rc != SSH_OK) {
    fprintf(stderr, "Error connecting to %s:%d: %s\n", hostname, port, ssh_get_error(my_ssh_session));
    ssh_free(my_ssh_session);
    exit(-1);
}

printf("Connected to %s. Verifying server identity...\n", hostname);

printf("Loading private key: %s\n", priv_key_path);
ssh_key priv_key;
rc = ssh_pki_import_privkey_file(priv_key_path, NULL, NULL, NULL, &priv_key);
if (rc != SSH_OK) {
    fprintf(stderr, "Error importing private key: %s\n", ssh_get_error(my_ssh_session));
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
    exit(-1);
}

printf("Authenticating with key...\n");
rc = ssh_userauth_publickey(my_ssh_session, NULL, priv_key);
ssh_key_free(priv_key);
if (rc != SSH_AUTH_SUCCESS) {
    fprintf(stderr, "Authentication failed: %s\n", ssh_get_error(my_ssh_session));
    ssh_disconnect(my_ssh_session);
    ssh_free(my_ssh_session);
    exit(-1);
}

printf("Authentication successful!\n");
```

Sau đó tuỳ vào tuỳ chọn là truyền file hay tải file, thì sẽ xử lý tạo phiên SCP để đọc hoặc ghi file lên server:

```c
int scp_upload_file(ssh_session session, const char *local_path, const char *remote_path) {
    struct stat file_stat;
    if (stat(local_path, &file_stat) != 0) {
        fprintf(stderr, "Error: Local file %s does not exist. Error: %s\n", local_path, strerror(errno));
        return -1;
    }

    transfer_stats_t stats;
    const char *filename = get_filename(local_path);
    strncpy(stats.filename, filename, sizeof(stats.filename) - 1);
    stats.filename[sizeof(stats.filename) - 1] = '\0';
    stats.filesize = file_stat.st_size;
    stats.bytes_transferred = 0;
    stats.start_time = time(NULL);

    ssh_scp scp = ssh_scp_new(session, SSH_SCP_WRITE, remote_path);
    if (scp == NULL) {
        fprintf(stderr, "Error creating SCP session: %s\n", ssh_get_error(session));
        return -1;
    }

    int rc = ssh_scp_init(scp);
    if (rc != SSH_OK) {
        fprintf(stderr, "Error initializing SCP: %s\n", ssh_get_error(session));
        ssh_scp_free(scp);
        return -1;
    }

    int local_fd = open(local_path, O_RDONLY);
    if (local_fd < 0) {
        fprintf(stderr, "Error opening local file %s: %s\n", local_path, strerror(errno));
        ssh_scp_free(scp);
        return -1;
    }
```

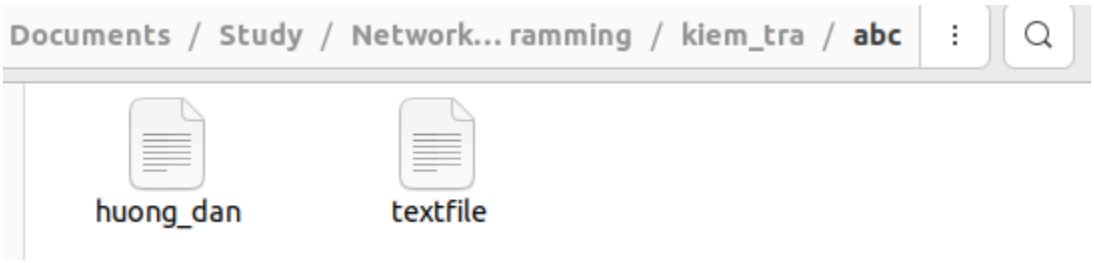
```c
int scp_download_file(ssh_session session, const char *remote_path, const char *local_path) {
    ssh_scp scp = ssh_scp_new(session, SSH_SCP_READ, remote_path);
    if (scp == NULL) {
        fprintf(stderr, "Error creating SCP session: %s\n", ssh_get_error(session));
        return -1;
    }

    int rc = ssh_scp_init(scp);
    if (rc != SSH_OK) {
        fprintf(stderr, "Error initializing SCP: %s\n", ssh_get_error(session));
        ssh_scp_free(scp);
        return -1;
    }

    rc = ssh_scp_pull_request(scp);
    if (rc != SSH_SCP_REQUEST_NEWFILE) {
        fprintf(stderr, "Error receiving file request: %s\n", ssh_get_error(session));
        ssh_scp_free(scp);
        return -1;
    }

    int file_size = ssh_scp_request_get_size(scp);
    const char *filename = ssh_scp_request_get_filename(scp);

    transfer_stats_t stats;
    strncpy(stats.filename, filename, sizeof(stats.filename) - 1);
    stats.filename[sizeof(stats.filename) - 1] = '\0';
    stats.filesize = file_size;
    stats.bytes_transferred = 0;
    stats.start_time = time(NULL);
```

Trước tiên với SSH server có sẵn của openssh-server, với người dùng nhoclahola đã được thiết lập khoá công khai lên server từ trước. Copy mã nguồn lên thư mục Downloads của người dùng nhoclahola.

```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra$ ./scp_key_client -u nhoclahola -h 192.16
8.119.132 -k ~/.ssh/id_rsa -s ./scp_key_client -d ~/Downloads
Connecting to nhoclahola@192.168.119.132:22...
Connected to 192.168.119.132. Verifying server identity...
Loading private key: /home/nhoclahola/.ssh/id_rsa
Authenticating with key...
Authentication successful!
Source is a file. Preparing to upload single file...
Starting upload: scp_key_client (36008 bytes) to /home/nhoclahola/Downloads
[scp_key_client] 100.0% (36008/36008 bytes) | 351.64 KB/s | 0s elapsed | ETA: 0s
Upload completed: scp_key_client (351.64 KB/s, 36008 bytes in 0.1 seconds)
Closing connection...
Connection closed.
```

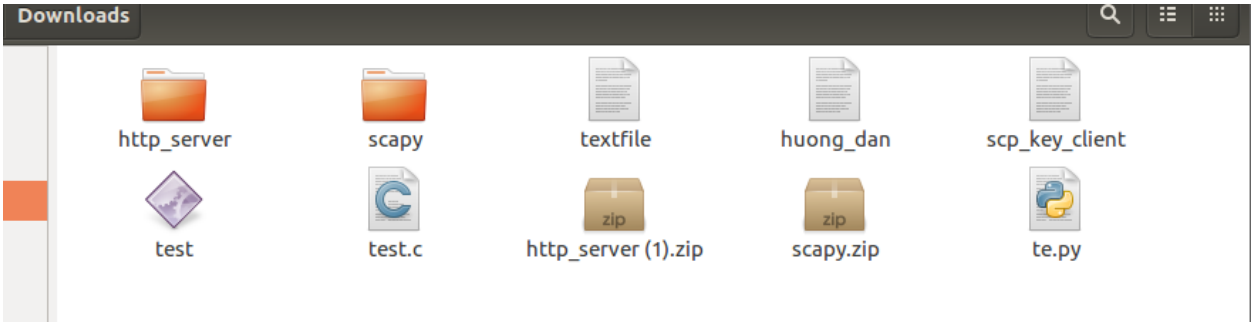
Có thể thấy từ log là người dùng đã được xác thực với khoá riêng tư thành công sau đó truyền file.

Tiếp theo sẽ thử truyền các file trong một thư mục abc:




```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra$ ./scp_key_client -u nhoclahola -h 192.16
8.119.132 -k ~/.ssh/id_rsa -s ./abc -d ~/Downloads
Connecting to nhoclahola@192.168.119.132:22...
Connected to 192.168.119.132. Verifying server identity...
Loading private key: /home/nhoclahola/.ssh/id_rsa
Authenticating with key...
Authentication successful!
Source is a directory. Preparing to upload multiple files...
Uploading files from directory: ./abc
Starting upload: huong_dan (107 bytes) to /home/nhoclahola/Downloads/huong_dan
[huong_dan] 100.0% (107/107 bytes) | 107.00 B/s | 1s elapsed | ETA: 0s
Upload completed: huong_dan (107.00 B/s, 107 bytes in 1.0 seconds)
Starting upload: textfile (0 bytes) to /home/nhoclahola/Downloads/textfile

Upload completed: textfile (0.00 B/s, 0 bytes in 0.1 seconds)

Directory upload completed: 2 files uploaded, 0 failed
Closing connection...
Connection closed.
```

Đã được truyền thành công.



Tương tự với tải file từ SSH server.

```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/kiem_tra$ ./scp_key_client -u nhoclahola -h 192.16
8.119.132 -k ~/.ssh/id_rsa -s ~/Downloads/test.c -d ~/Downloads --download
Connecting to nhoclahola@192.168.119.132:22...
Connected to 192.168.119.132. Verifying server identity...
Loading private key: /home/nhoclahola/.ssh/id_rsa
Authenticating with key...
Authentication successful!
Source is a remote file. Preparing to download single file...
Starting download: test.c (441 bytes) to /home/nhoclahola/Downloads/test.c
[test.c] 100.0% (441/441 bytes) | 4.31 KB/s | 0s elapsed | ETA: 0sError reading from remote file: ssh_scp_read c
alled under invalid state
Closing connection...
Connection closed.
```

Như vậy là code hoạt động như mong đợi.

5.2. SCP server

Tạo hostkey tại server để client xác thực server này:

```
nhoclahola@ubuntu:~/Documents/kiem_Tra$ ssh-keygen -t rsa -f ssh_host_rsa_key
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in ssh_host_rsa_key.
Your public key has been saved in ssh_host_rsa_key.pub.
The key fingerprint is:
SHA256:O413qaHMpxrFApbl1OfQHxL+pAonmLWhAtDPeivAk/8 nhoclahola@ubuntu
The key's randomart image is:
+---[RSA 2048]----+
|..     o. ...    |
|. .   =  o.+ .   |
|.  o+ +  +.o..   |
| . .o* +  .+.    |
|. o.+ = S . .    |
|.+...  * =   .   |
| .o. .. = + o    |
|  ...  + +.+     |
|   ..E..=o.      |
+----[SHA256]-----+
```

```
nhoclahola@ubuntu:~/Documents/kiem_Tra$ ./scp_server
SCP Server listening on port 2222
```

- Đầu tiên, server khởi động bằng cách dùng hàm ssh_bind_listen để chờ kết nối SSH tới cổng chỉ định (mặc định là 2222). Server liên tục lặp lại việc chấp nhận kết nối (ssh_bind_accept) khi có client mới.

```
while (!authenticated && auth_attempts < 3) {
    message = ssh_message_get(session);
    if (message == NULL) {
        fprintf(stderr, "[ERROR] No auth message received\n");
        break;
    }

    if (ssh_message_type(message) == SSH_REQUEST_AUTH) {
        switch (ssh_message_subtype(message)) {
            case SSH_AUTH_METHOD_PASSWORD:
                printf("[INFO] Password auth attempt from %s (user: %s)\n",
                       client_ip, ssh_message_auth_user(message));
                authenticated = 1;
                ssh_message_auth_reply_success(message, 0);
                break;

            case SSH_AUTH_METHOD_PUBLICKEY:
                printf("[INFO] Publickey auth attempt from %s (user: %s)\n",
                       client_ip, ssh_message_auth_user(message));
                authenticated = 1;
                ssh_message_auth_reply_success(message, 0);
                break;

            default:
                ssh_message_auth_set_methods(message, SSH_AUTH_METHOD_PASSWORD | SSH_AUTH_METHOD_PUBLICKEY);
                ssh_message_reply_default(message);
                break;
        }
    } else {
        ssh_message_reply_default(message);
    }
```

- Khi client kết nối tới, server sẽ lấy thông tin IP của client và thực hiện trao đổi khóa SSH (ssh_handle_key_exchange) để thiết lập kênh bảo mật.

```
while (!authenticated && auth_attempts < 3) {
    message = ssh_message_get(session);
    if (message == NULL) {
        fprintf(stderr, "[ERROR] No auth message received\n");
        break;
    }

    if (ssh_message_type(message) == SSH_REQUEST_AUTH) {
        switch (ssh_message_subtype(message)) {
            case SSH_AUTH_METHOD_PASSWORD:
                printf("[INFO] Password auth attempt from %s (user: %s)\n",
                       client_ip, ssh_message_auth_user(message));
                authenticated = 1;
                ssh_message_auth_reply_success(message, 0);
                break;

            case SSH_AUTH_METHOD_PUBLICKEY:
                printf("[INFO] Publickey auth attempt from %s (user: %s)\n",
                       client_ip, ssh_message_auth_user(message));
                authenticated = 1;
                ssh_message_auth_reply_success(message, 0);
                break;

            default:
                ssh_message_auth_set_methods(message, SSH_AUTH_METHOD_PASSWORD | SSH_AUTH_METHOD_PUBLICKEY);
                ssh_message_reply_default(message);
                break;
        }
    } else {
        ssh_message_reply_default(message);
    }
```

- Kiểm tra lệnh SCP nhận được: Nếu bắt đầu bằng scp -t, nghĩa là client muốn upload file, sẽ chuyển xử lý qua handle_scp_upload, ngược lại với -f thì là download.

```
int handle_channel(ssh_session session, ssh_channel channel, const char *command) {
    char response[16] = {0};

    if (strncmp(command, "test -d", 7) == 0) {
        char path[512] = {0};
        if (sscanf(command, "test -d %511s", path) == 1) {
            struct stat st;
            if (stat(path, &st) == 0 && S_ISDIR(st.st_mode)) {
                strcpy(response, "DIR\n");
            } else {
                strcpy(response, "FILE\n");
            }
            ssh_channel_write(channel, response, strlen(response));

            // Gửi EOF và đóng channel sau khi trả lời
            ssh_channel_send_eof(channel);
            ssh_channel_close(channel);
            return SSH_OK;
        } else {
            strcpy(response, "FILE\n");
            ssh_channel_write(channel, response, strlen(response));

            // Gửi EOF và đóng channel
            ssh_channel_send_eof(channel);
            ssh_channel_close(channel);
            return SSH_OK;
        }
    }
    else if (strncmp(command, "scp -t", 6) == 0) {
        return handle_scp_upload(session, channel, command);
    }
    else if (strncmp(command, "scp -f", 6) == 0) {
        return handle_scp_download(session, channel, command);
    }
```

- Nhận và ghi file: Server liên tục đọc dữ liệu từ channel SSH và ghi vào file. Trong quá trình này, liên tục cập nhật tiến độ truyền file.

```c
int handle_scp_upload(ssh_session session, ssh_channel channel, const char *command, const char *client_ip) {
    char timestamp[32];
    get_timestamp(timestamp, sizeof(timestamp));
    printf("[%s] [INFO] SCP upload command received from %s: %s\n", timestamp, client_ip, command);

    char destination[512] = {0};
    if (sscanf(command, "scp -t %511s", destination) != 1) {
        fprintf(stderr, "[%s] [ERROR] Invalid SCP upload command: %s\n", timestamp, command);
        return SSH_ERROR;
    }

    struct stat st = {0};
    if (stat(destination, &st) == -1) {
        if (mkdir(destination, 0755) == -1 && errno != EEXIST) {
            fprintf(stderr, "[%s] [ERROR] Failed to create directory %s: %s\n", timestamp, destination, strerror(errno));
            return SSH_ERROR;
        }
    }

    char ok = 0;
    ssh_channel_write(channel, &ok, 1);

    file_transfer_info transfer_info = {0};
    strncpy(transfer_info.destination, destination, sizeof(transfer_info.destination) - 1);
    strncpy(transfer_info.client_ip, client_ip, sizeof(transfer_info.client_ip) - 1);

    char buffer[BUFFER_SIZE];
    int nbytes = ssh_channel_read(channel, buffer, BUFFER_SIZE, 0);
    if (nbytes <= 0) {
        fprintf(stderr, "[%s] [ERROR] Failed to read file header: %s\n", timestamp, ssh_get_error(session));
        return SSH_ERROR;
    }
```

```
nhoclahola@nhoclahola-vm:~/Documents/Study/NetworkProgramming/ktem_tra$ ./scp_key_client -u nhoclahola -h 192.16
8.119.132 -k ~/.ssh/id_rsa -s huong_dan2 -d ~/Downloads -p 2222
Connecting to nhoclahola@192.168.119.132:2222...
Connected to 192.168.119.132. Verifying server identity...
Loading private key: /home/nhoclahola/.ssh/id_rsa
Authenticating with key...
Authentication successful!
Source is a file. Preparing to upload single file...
Starting upload: huong_dan2 (107 bytes) to /home/nhoclahola/Downloads
Error pushing file: Remote channel is closed
```