I. Chuẩn bị

a. Cài đặt openssl

```
buiducthang@ubuntu:~$ sudo apt-get install openssl
```

b. Cấu hình Openssl

Sử dụng lệnh sudo vi /usr/lib/ssl/openssl.cnf.

```
# testoid1=1.2.3.4
# Or use config file substitution like this:
# testoid2=${testoid1}.5.6

# Policies used by the TSA examples.
tsa_policy1 = 1.2.3.4.1
tsa_policy2 = 1.2.3.4.5.6
tsa_policy3 = 1.2.3.4.5.7

####################################################################
[ ca ]
default_ca      = CA_default            # The default ca section

####################################################################
[ CA_default ]

dir             = /root         # Where everything is kept
certs           = $dir          # Where the issued certs are kept
crl_dir         = $dir/crl              # Where the issued crl are kept
database        = $dir/index.txt        # database index file.
#unique_subject = no                    # Set to 'no' to allow creation of
                                        # several ctificates with same subject.
new_certs_dir   = $dir          # default place for new certs.

                                                            49,0-1          7%
```

Trong đó dir là đường dẫn thư mục, certs là đường dẫn tới thư mục chứa chứng chỉ, crl_dir là đường dẫn tới crl, database là nơi lưu trữ file database.

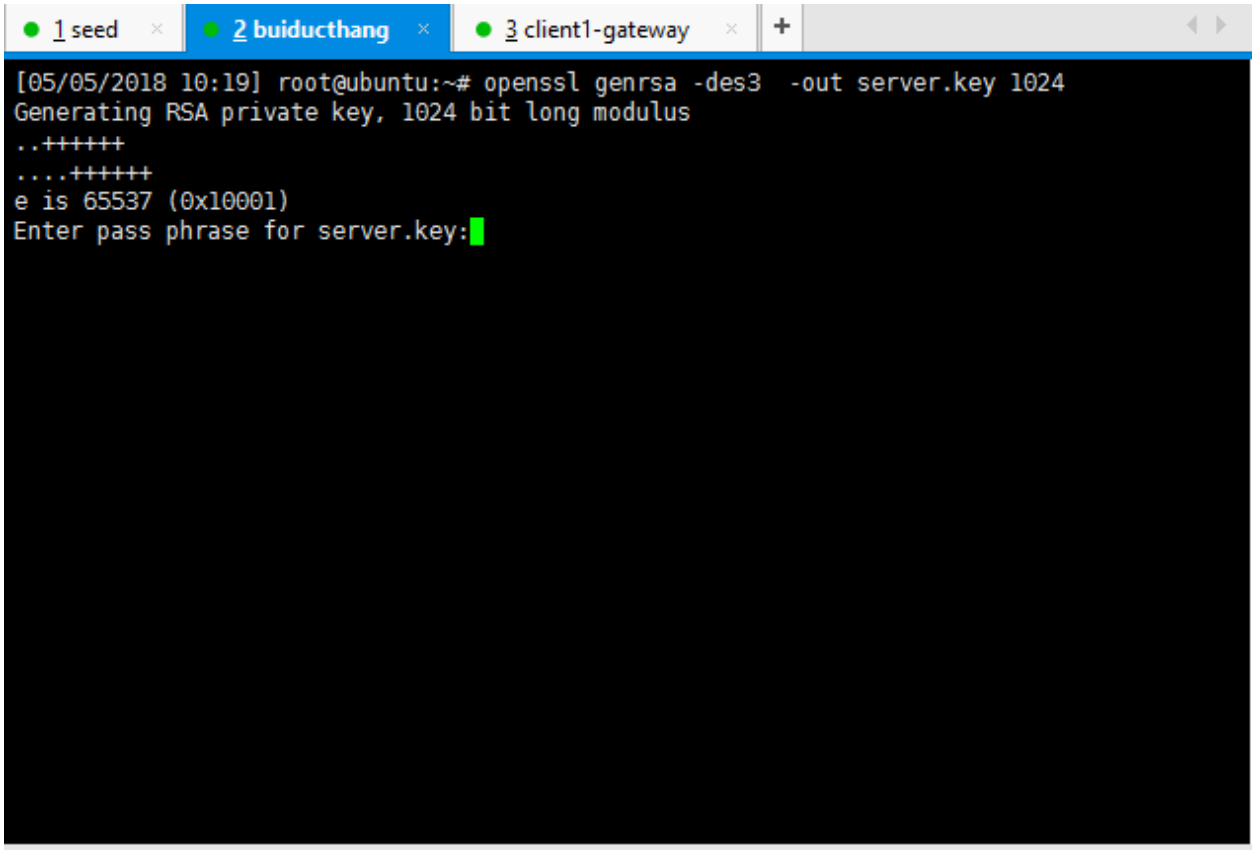
c. Tạo CA

```
[05/05/2018 10:15] root@ubuntu:~# openssl req -new -x509 -keyout ca.key -out ca.crt -con
fig /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
...++++++
.....................++++++
writing new private key to 'ca.key'
Enter PEM pass phrase:
```

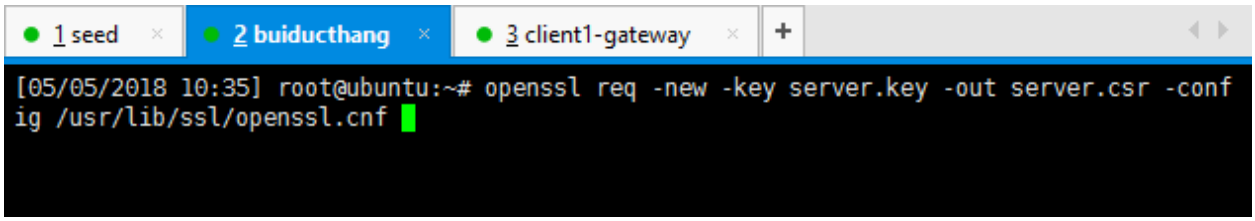
Điền mật khẩu. Chú ý cần ghi nhớ mật khẩu này. Sau đó điền các thông tin tiếp theo

```
[05/05/2018 10:15] root@ubuntu:~# openssl req -new -x509 -keyout ca.key -out ca.crt -con
fig /usr/lib/ssl/openssl.cnf
Generating a 1024 bit RSA private key
...++++++
.....................++++++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:VN
State or Province Name (full name) [Some-State]:ptit
Locality Name (eg, city) []:ptit
Organization Name (eg, company) [Internet Widgits Pty Ltd]:
```

d. Sinh khóa cho server

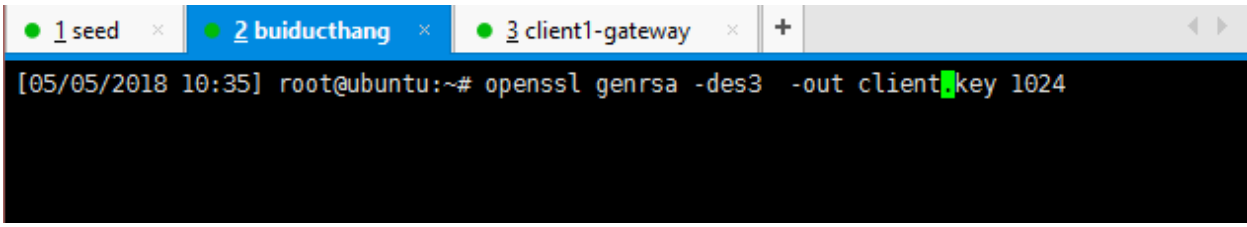


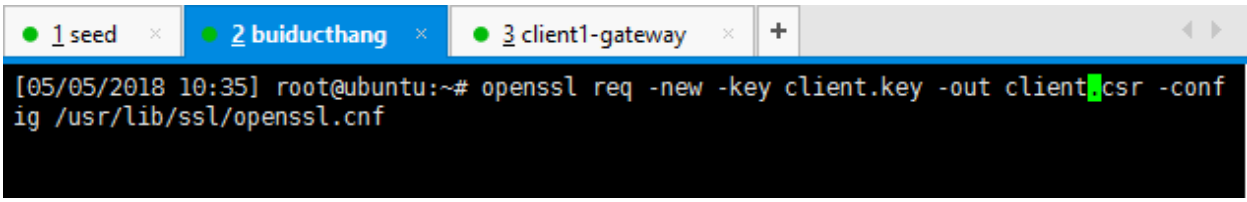
Điền mật khẩu



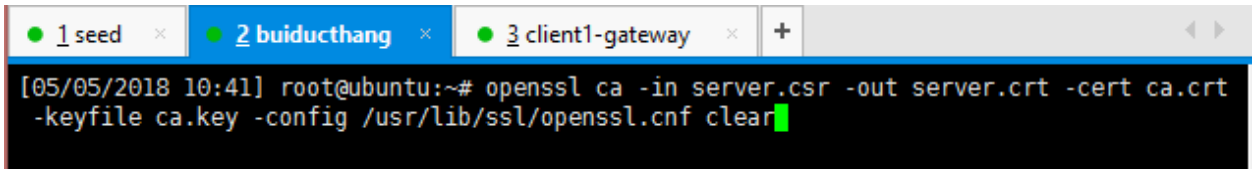
e. Tạo khóa cho client

Làm tương tự với server

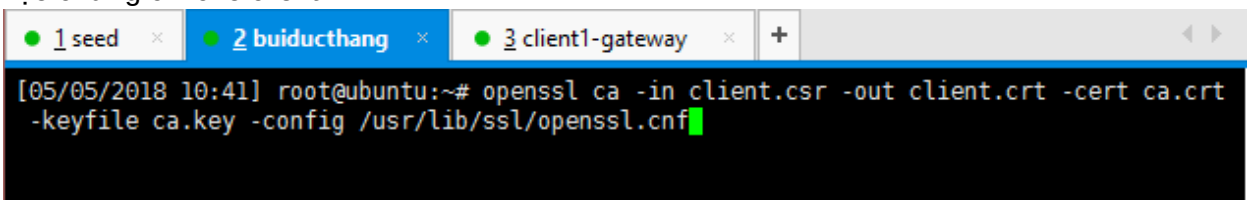




f. Tạo chứng chỉ cho server



g. Tạo chứng chỉ cho client

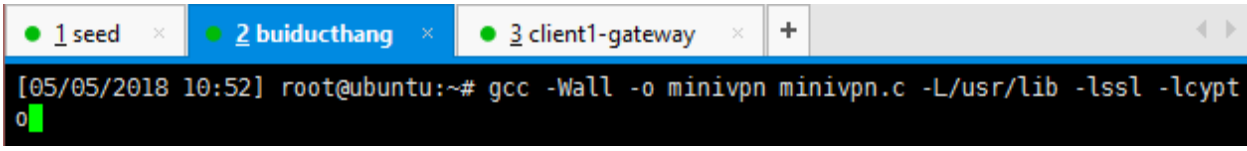


h.

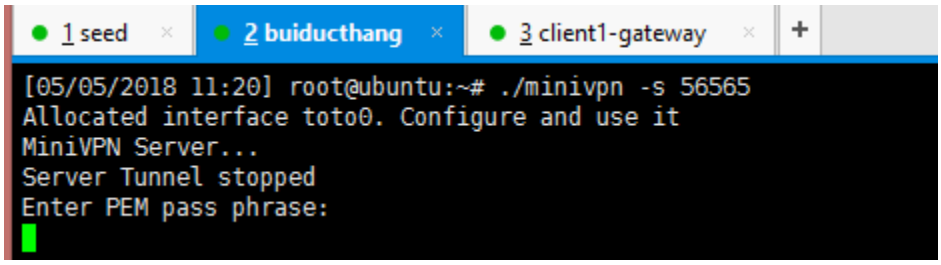
II. Host to Host

a. Cài đặt 2 máy ảo. Chọn 1 máy làm server, 1 máy làm client. Ví dụ server có IP là: 192.168.32.138, client có IP là : 192.168.32.129

b. Download minivpn.c và Compile



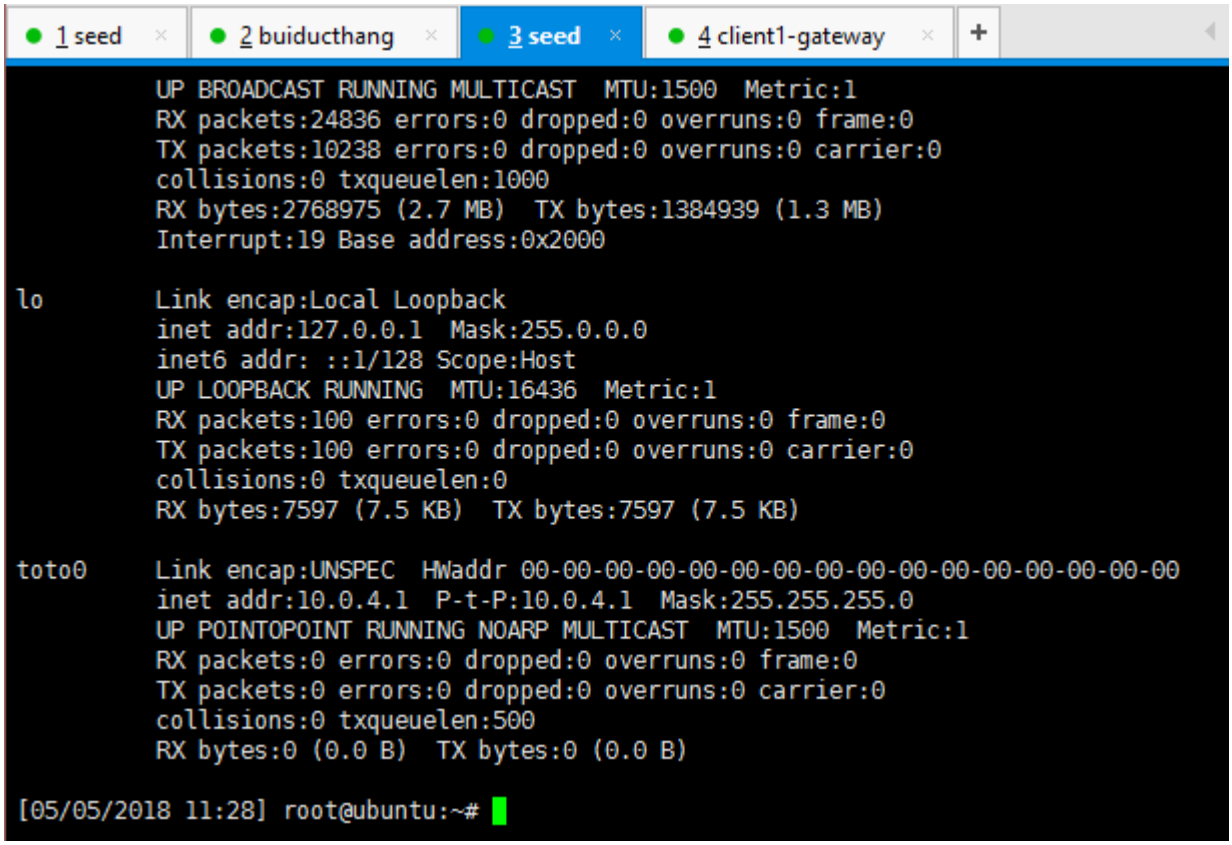
c. Tại máy server có IP 192.168.32.138 ta chạy lệnh sau:



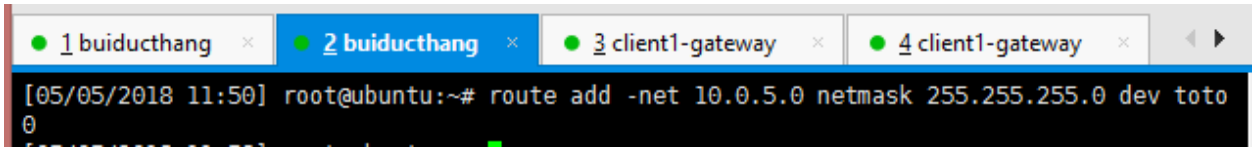
PEM pass là nhập mật khẩu của chứng chỉ vừa tạo.

Tiếp theo chạy các lệnh sau để gán địa chỉ cho card mạng và bật card mạng

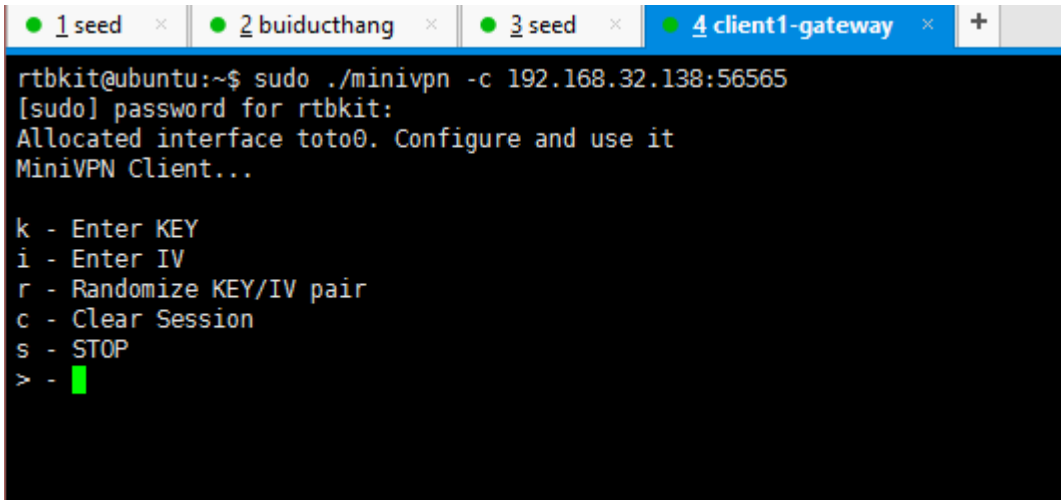
```
[05/05/2018 11:27] root@ubuntu:~# ip addr add 10.0.4.1/24 dev toto0
[05/05/2018 11:28] root@ubuntu:~# ifconfig toto0 up
```



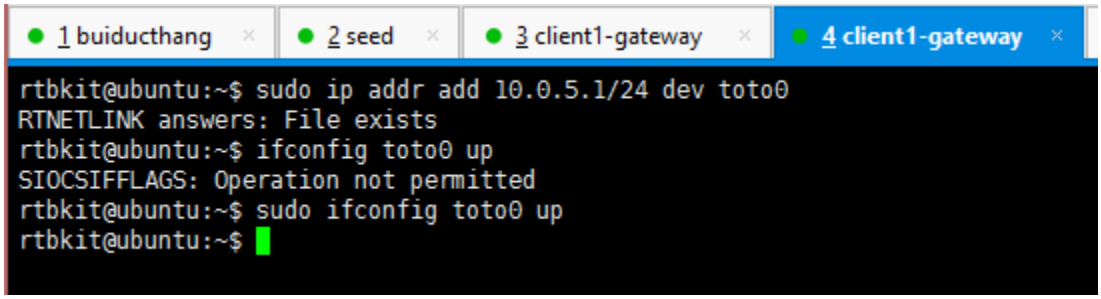
Bật card mạng thành công



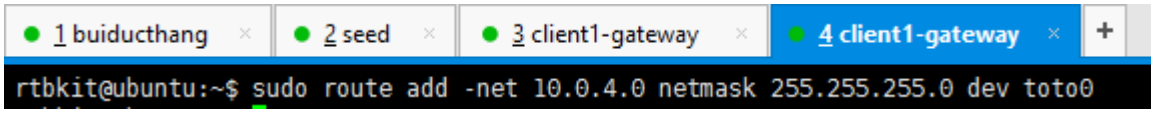
d. Tại máy client có IP là 192.168.32.129 ta chạy lệnh sau:



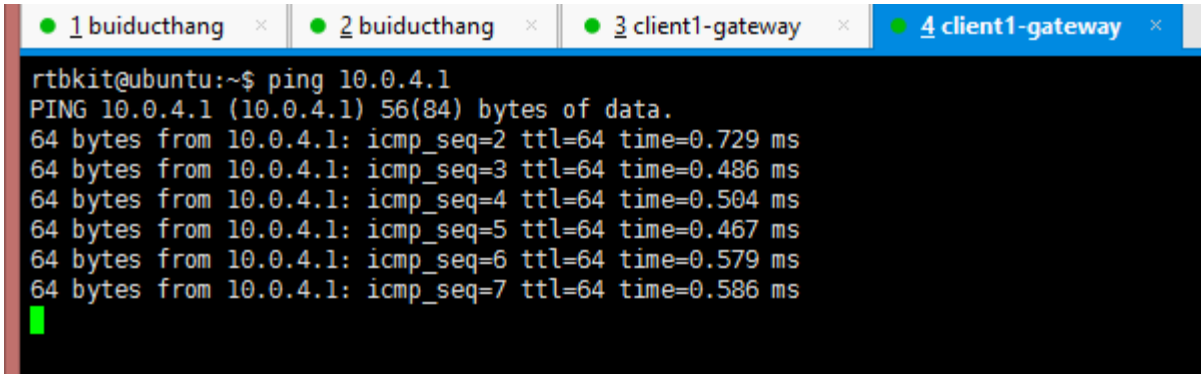
Chọn các sinh khóa



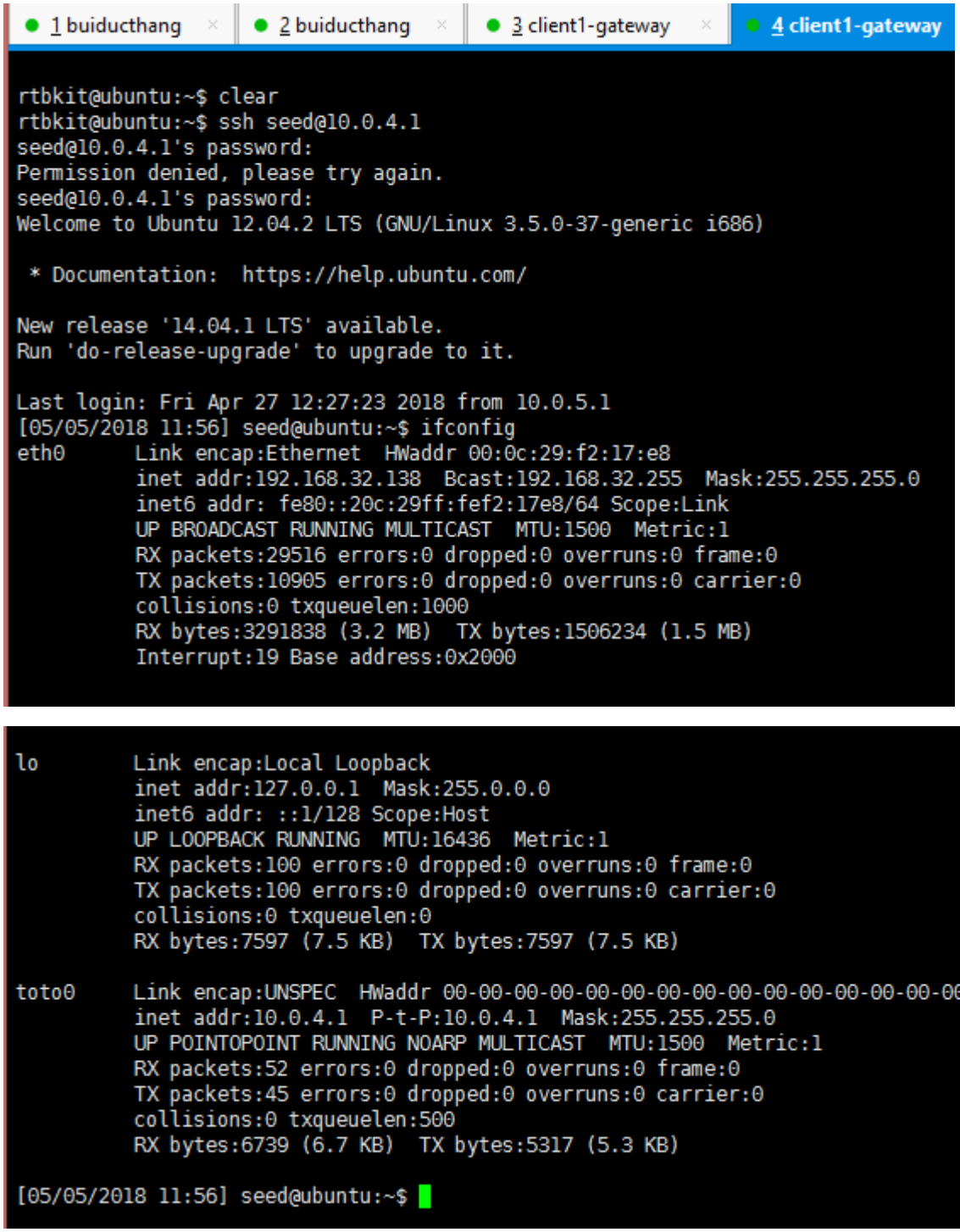
Gán địa chỉ ip cho card mạng và bật card mạng



Kiểm tra tunning thành công bằng cách ping đến máy server



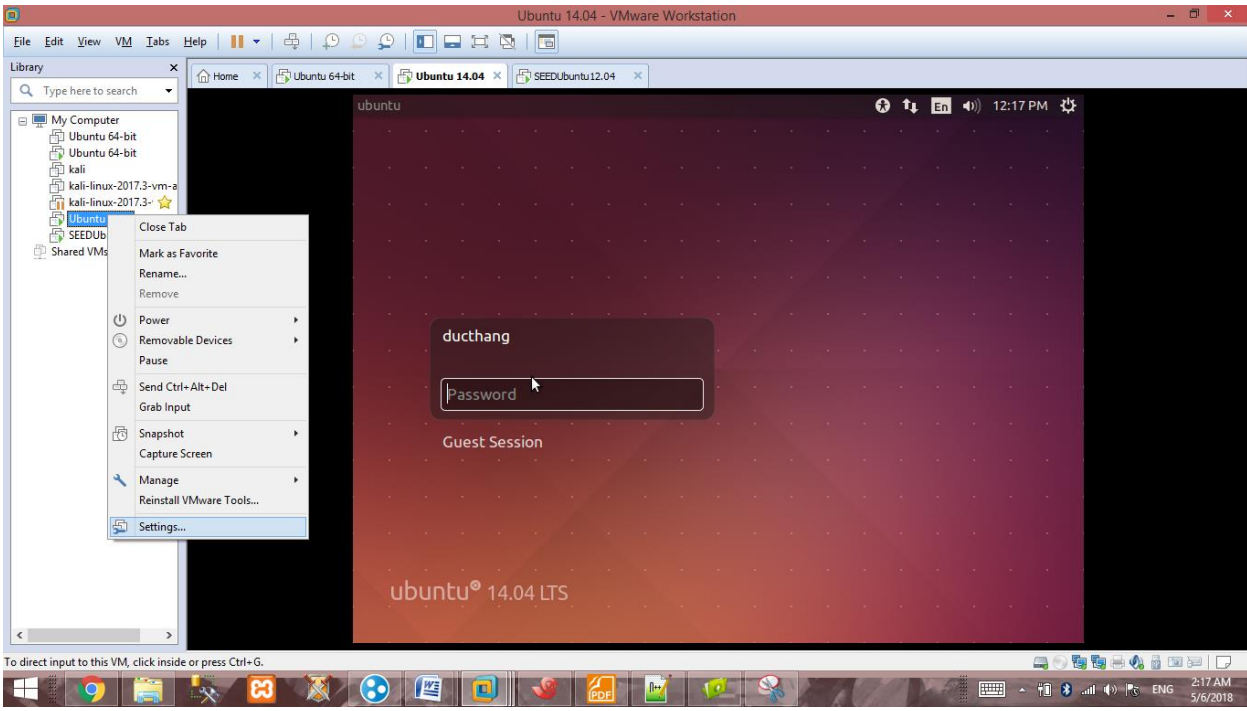
Hoặc ssh



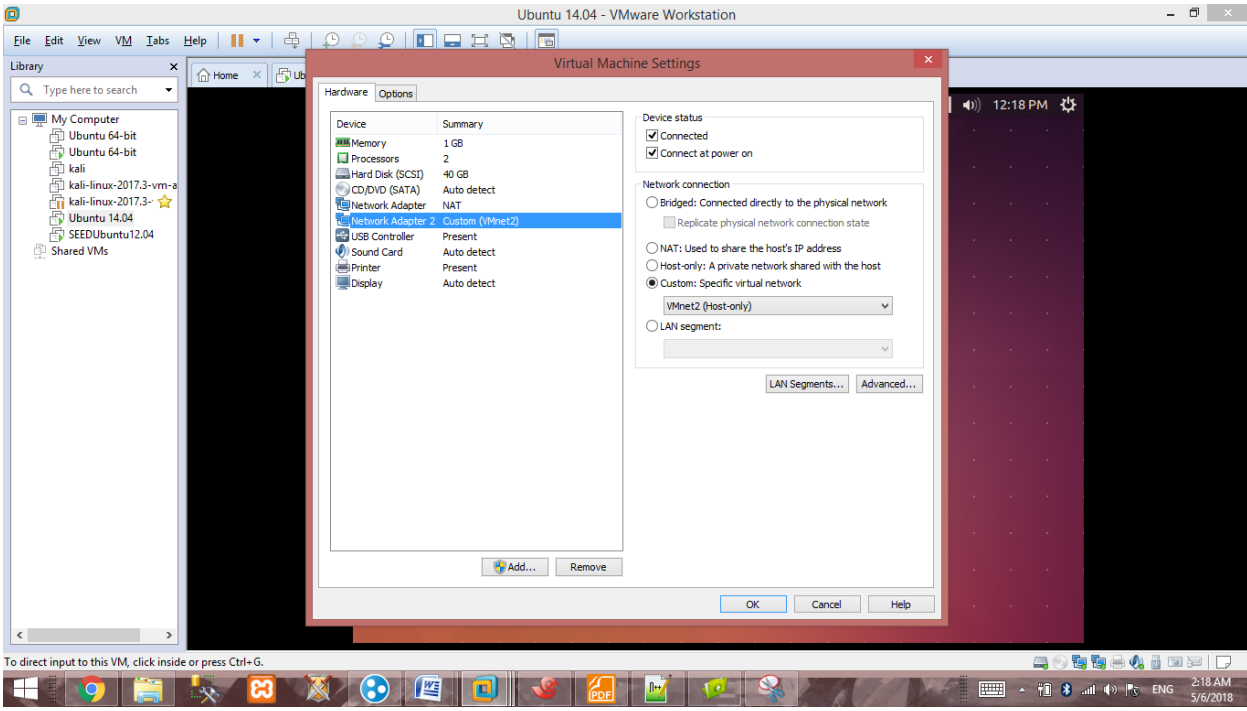
III. Host to Gateway

a. Setup private network bằng Vmware

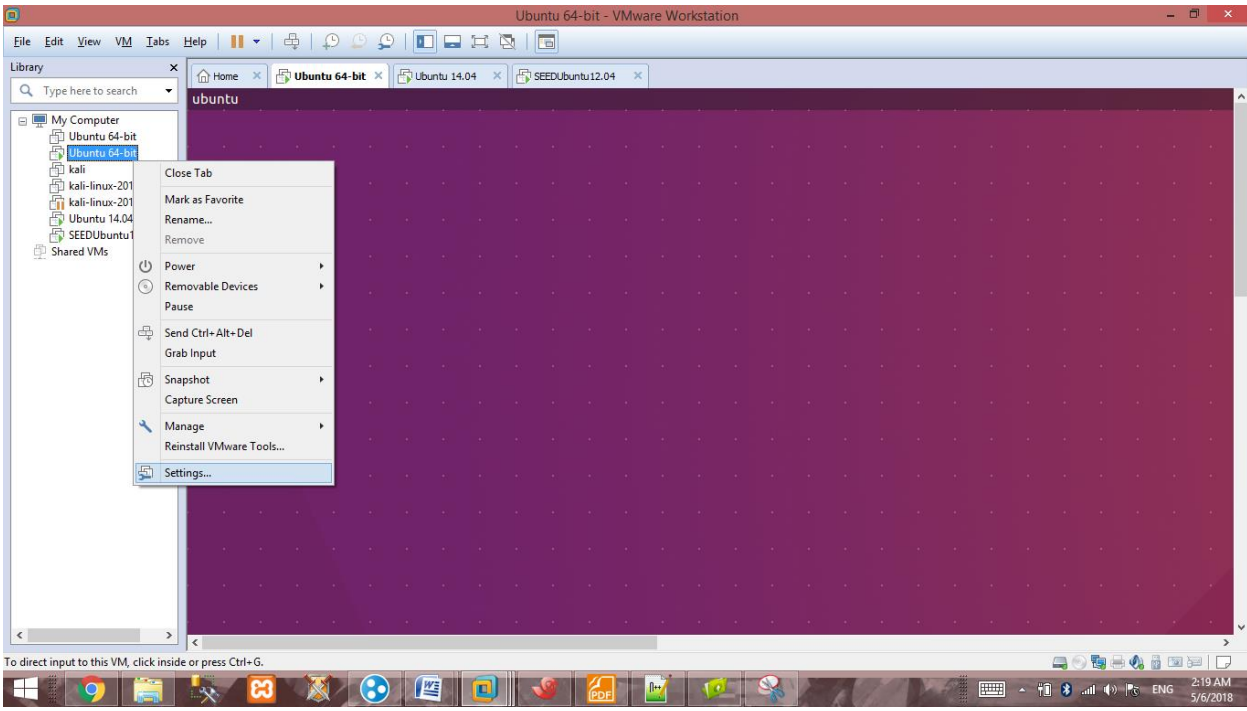
Làm như hình ảnh sau:

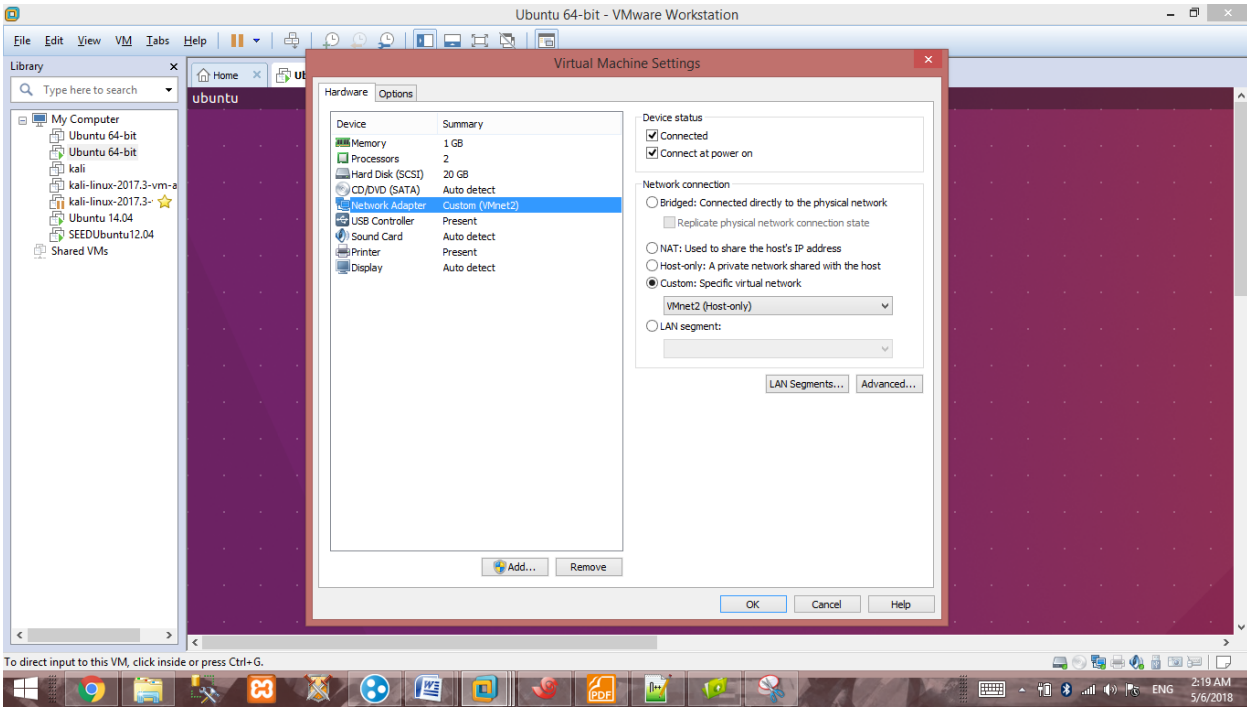


Thêm một card mạng mới cho máy được chọn làm gateway. 1 card mạng sử dụng NAT, card mạng còn lại sử dụng host-only

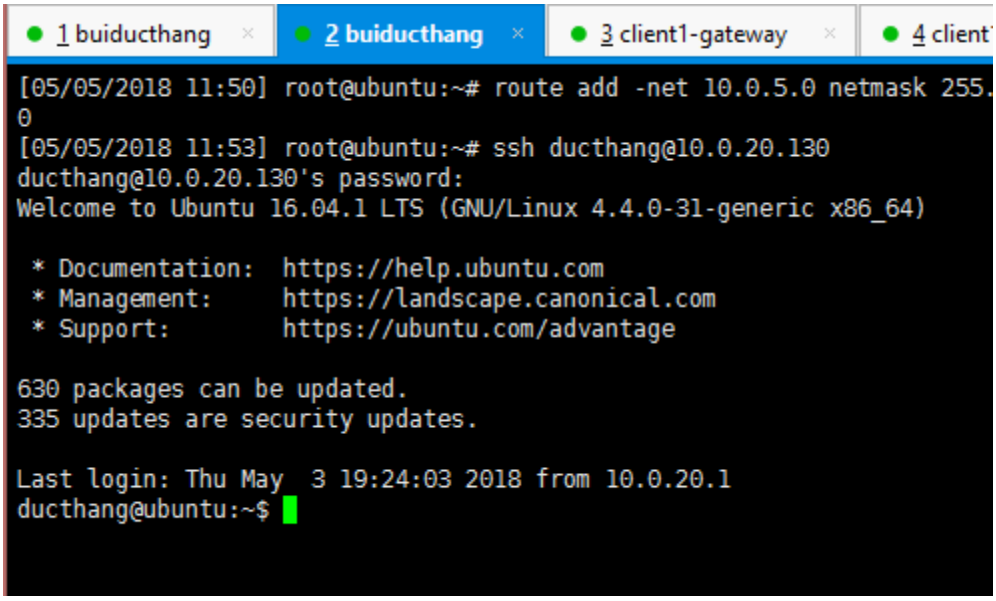


Đến máy trong mạng private





b. Cài đặt miniVPN trên máy chọn làm gateway như ở phần I
c. Kiểm tra
   Ping hoặc SSH thử đến máy trong mạng private (Không phải gateway) từ một máy ngoài private network



IV. VPN

Mã hóa bằng thư viện Openssl:

```
void do_encrypt(char *in, int inl, char *out, int *outl)
{
    EVP_CIPHER_CTX ctx;
    int tmpl = 0;

    if (DEBUG) write(1,"0", 1);
    EVP_CIPHER_CTX_init(&ctx);
    if (DEBUG) write(1,"1", 1);
    if(0 == EVP_EncryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL, KEY, IV)) PERROR("EVP_EncryptInit_ex");
    if (DEBUG) write(1,"2", 1);
    if(0 == EVP_EncryptUpdate(&ctx, out, outl, in, inl))                PERROR("EVP_EncryptUpdate");
    if (DEBUG) write(1,"3", 1);
    if(0 == EVP_EncryptFinal_ex(&ctx, out+*outl, &tmpl))                PERROR("EVP_EncryptFinal_ex");
    *outl += tmpl;
    if (DEBUG) write(1,"4", 1);
    EVP_CIPHER_CTX_cleanup(&ctx);
    if (DEBUG) write(1,"5", 1);
}
```

Giải mã:

```
void do_decrypt(char *in, int inl, char *out, int *outl)
{
    EVP_CIPHER_CTX ctx;
    int tmpl = 0;

    if (DEBUG) write(1,"6", 1);
    EVP_CIPHER_CTX_init(&ctx);
    if (DEBUG) write(1,"7", 1);
    if(0 == EVP_DecryptInit_ex(&ctx, EVP_aes_128_cbc(), NULL, KEY, IV)) PERROR("EVP_DecryptInit_ex");
    if (DEBUG) write(1,"8", 1);
    if(0 == EVP_DecryptUpdate(&ctx, out, outl, in, inl))                PERROR("EVP_DecryptUpdate");
    if (DEBUG) write(1,"9", 1);
    if(0 == EVP_DecryptFinal_ex(&ctx, out+*outl, &tmpl))                PERROR("EVP_DecryptFinal_ex");
    *outl += tmpl;
    if (DEBUG) write(1,"a", 1);
    EVP_CIPHER_CTX_cleanup(&ctx);
    if (DEBUG) write(1,"b", 1);
}
```

V. Authentication and Key Exchange

```
/* PKI: Authenticate the Server */
if (!ctx || !ssl || (sd < 0)) {
    if (DEBUG) printf("PKI: Authenticating the Server...\n");
    SSLeay_add_ssl_algorithms();
    SSL_load_error_strings();
    ctx = SSL_CTX_new(meth);
    if (NULL == ctx) {if (DEBUG) printf("ctx error\n");err=-1;continue;};

    SSL_CTX_set_verify(ctx,SSL_VERIFY_PEER,NULL);
    SSL_CTX_load_verify_locations(ctx,CCACERT,NULL);

    if (SSL_CTX_use_certificate_file(ctx,CCERTF,SSL_FILETYPE_PEM)<=0){
       ERR_print_errors_fp(stderr);err=-2;continue;}
    if (SSL_CTX_use_PrivateKey_file(ctx,CKEYF,SSL_FILETYPE_PEM)  <=0){
       ERR_print_errors_fp(stderr);err=-3;continue;}
    if (!SSL_CTX_check_private_key(ctx)){
       printf("Private key doesn't match cert. public key\n");err=-4;continue;}

    /* Create a socket and connect to server using normal socket calls. */
    sd = socket (AF_INET, SOCK_STREAM, 0);
    if (-1 == sd) {perror("socket");err=-5;continue;}

    memset(&sa,0,sizeof(sa));
    sa.sin_family = AF_INET;
    inet_aton(ip, &sa.sin_addr); /* Server IP */
    sa.sin_port   = htons(port); /* Server Port */

    err = connect(sd,(struct sockaddr*)&sa,sizeof(sa));
    if (-1 == err) {perror("connect");err=-6;continue;}
```

```
/* DATA EXCHANGE AREA */
if (ssl) {
   /* Write CommonName (CN) to Server */
   if (DEBUG) printf("Writing CN [%s] to Server\n",szCommonName);
   strcpy(msg,"CN:");
   strcat(msg,szCommonName);
   err = SSL_write(ssl,msg,strlen(msg));  if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
   /* Read CN reply from Server */
   err = SSL_read(ssl,msg,sizeof(msg)-1); if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
   msg[err] = '\0';
   if (DEBUG) printf("Server replied with %d chars:'%s'\n",err,msg);
   if (strncmp(msg,"CN_ACK",6)) {cnValid=0;printf("Client_CN_ACK_Error\n");}
   else cnValid=1;

   /* Write KEY to Server */
   if (DEBUG) printf("Writing KEY [%s] to Server\n",KEY);
   strcpy(msg,"KEY:");
   strcat(msg,KEY);
   err = SSL_write(ssl,msg,strlen(msg));  if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
   /* Read KEY reply from Server */
   err = SSL_read(ssl,msg,sizeof(msg)-1); if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
   msg[err] = '\0';
   if (DEBUG) printf("Server replied with %d chars:'%s'\n",err,msg);
   if (strncmp(msg,"KEY_ACK",7)) {keyValid=0;printf("Client_KEY_ACK_Error\n");}
   else keyValid=1;

   /* Write IV to Server */
   if (DEBUG) {printf("Writing IV ["); dumpBuf(IV,IV_LENGTH); printf("] to Server\n");}
   strcpy(msg,"IV:");
   memcpy(&msg[3],&IV,sizeof(IV));
   err = SSL_write(ssl,msg,3+sizeof(IV)); if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
   /* Read IV reply from Server */
   err = SSL_read(ssl,msg,sizeof(msg)-1); if (-1 == err) {ERR_print_errors_fp(stderr); continue;}
```

```c
/* PKI: Authenticate the Client */
if (!ctx || !ssl || (sd < 0)) {
   if (DEBUG) printf("PKI: Authenticating the Client...\n");

   /* SSL preliminaries. We keep the certificate and key with the context. */
   SSLeay_add_ssl_algorithms();
   SSL_load_error_strings();
   ctx = SSL_CTX_new (meth);
   if (!ctx) {ERR_print_errors_fp(stderr);err=-1;continue;}

   SSL_CTX_set_verify(ctx,SSL_VERIFY_PEER,NULL); /* whether verify the certificate */
   SSL_CTX_load_verify_locations(ctx,SCACERT,NULL);

   if (SSL_CTX_use_certificate_file(ctx,SCERTF,SSL_FILETYPE_PEM)<=0){
      ERR_print_errors_fp(stderr);err=-2;continue;}
   if (SSL_CTX_use_PrivateKey_file(ctx,SKEYF,SSL_FILETYPE_PEM)  <=0){
      ERR_print_errors_fp(stderr);err=-3;continue;}
   if (!SSL_CTX_check_private_key(ctx)){
      fprintf(stderr,"Private key does not match the certificate public key\n");err=-4;continue;}

   /* Prepare TCP socket for receiving connections */
   //listen_sd = socket(AF_INET,SOCK_STREAM,0);
   if (-1 == listen_sd) {perror("socket");err=-5;continue;}

   //memset(&sa_serv,0,sizeof(sa_serv));
   //sa_serv.sin_family      = AF_INET;
   //sa_serv.sin_addr.s_addr = INADDR_ANY;
   //sa_serv.sin_port        = htons(PORT); /* Server Port number */

   //err = bind(listen_sd,(struct sockaddr*)&sa_serv,sizeof(sa_serv));
   //if (-1 == err) {perror("bind");err=-6;continue;}

   /* Receive a TCP connection. */
   err = listen(listen_sd,5);
   if (-1 == err) {perror("listen");err=-7;continue;}
```

```c
/* DATA EXCHANGE - Receive messages and send replies */
if (ssl) {
    /* Read CN from Client */
    err = SSL_read(ssl,msg,sizeof(msg)-1);
    if (-1 == err) {ERR_print_errors_fp(stderr);err=-14;continue;}
    msg[err] = '\0';
    if (DEBUG) printf("Received from Client %d chars:'%s'\n",err,msg);
    /* Audit CN from Client */
    cnValid = 0;
    if (!strncmp(msg,"CN:",3)) {
        char *pch_start = strstr(subjectstr,"/CN=");
        if (NULL != pch_start) {
            char *pch_end = strstr(&pch_start[1],"/");
            if (NULL != pch_end) {
                int cn_len = (int)pch_end - (int)&pch_start[4];
                if (!strncmp(&pch_start[4],&msg[3],cn_len)) {
                    cnValid=1;
                }//if
            }//if
        }//if
    }//if
    /* Write CN reply to Client */
    if (1 == cnValid) {
        err = SSL_write(ssl,"CN_ACK",strlen("CN_ACK"));
        if (-1 == err) {ERR_print_errors_fp(stderr);err=-15;continue;}
    }//if
    else {
        err = SSL_write(ssl,"CN_NACK",strlen("CN_NACK"));
        if (-1 == err) {ERR_print_errors_fp(stderr);err=-16;continue;}
    }//else

    /* Read KEY from Client */
    err = SSL_read(ssl,msg,sizeof(msg)-1);
    if (-1 == err) {ERR_print_errors_fp(stderr);err=-17;continue;}
    msg[err] = '\0';
    if (DEBUG) printf("Received from Client %d chars:'%s'\n",err,msg);
```

```
/* Read KEY from Client */
err = SSL_read(ssl,msg,sizeof(msg)-1);
if (-1 == err) {ERR_print_errors_fp(stderr);err=-17;continue;}
msg[err] = '\0';
if (DEBUG) printf("Received from Client %d chars:'%s'\n",err,msg);
/* Write KEY reply to Client */
if (!strncmp(msg,"KEY:",4)) {
   strncpy(KEY,&msg[4],MAX_KEY_LENGTH);
   keyValid=1;
   err = SSL_write(ssl,"KEY_ACK",strlen("KEY_ACK"));
   if (-1 == err) {ERR_print_errors_fp(stderr);err=-18;continue;}
}//if
else {
   keyValid=0;
   err = SSL_write(ssl,"KEY_NACK",strlen("KEY_NACK"));
   if (-1 == err) {ERR_print_errors_fp(stderr);err=-19;continue;}
}//else

/* Read IV from Client */
err = SSL_read(ssl,msg,sizeof(msg)-1);
if (-1 == err) {ERR_print_errors_fp(stderr);err=-20;continue;}
msg[err] = '\0';
if (DEBUG) {printf("Received from Client %d chars:'%c%c%c",err,msg[0],msg[1],msg[2]);
          dumpBuf(&msg[3],err-3);printf("'\n");}
/* Write IV reply to Client */
if (!strncmp(msg,"IV:",3)) {
   memcpy(IV,&msg[3],IV_LENGTH);
   ivValid=1;
   err = SSL_write(ssl,"IV_ACK",strlen("IV_ACK"));
   if (-1 == err) {ERR_print_errors_fp(stderr);err=-21;continue;}
}//if
else {
   ivValid=0;
   err = SSL_write(ssl,"IV_NACK",strlen("IV_NACK"));
   if (-1 == err) {ERR_print_errors_fp(stderr);err=-22;continue;}
}//else
```