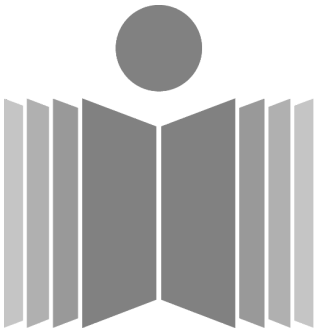


భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Assignment6 - Report

April 06, 2022

Network Security

CS6903

Team

Roll No.	Name
CS19BTECH11012	Nisha M
CS21MTECH14008	Koustav Choudhury
CS22MTECH02003	Priyansha Tiwari

Task 1:

Root certificate:

- **Command to create 512 bit ECC key-**
  openssl ecparam -name brainpoolP512t1 -genkey -noout -out rootkey_ecc.pem

```
EC private Key -
-----BEGIN EC PRIVATE KEY-----
MIHaAgEBBEBaPTP/Q+LB37Sy2itdT1iOah2yTDETJCnK6nwyV8/eO54R5ad1TLZw
iXyvJiBic3WRHnWMPXUrSEIkjlPtRz0KoAsGCSskAwMCCAEBDqGBhQOBggAEUNLt
Adg9uwtXqGnQQYNTmt4JvHUhgIICFDCB3qPVwd1kwzfHJd/dqCWmKReAtXmxAMQ/
1+xMJe6RwLUtAjht0yn1A7fiMtMoe/mubJeiw3lA6KaSN6dqNFICln6xQoQ8f++2
B8oOhUPUSZ2jFLlO9kQ/AKOFI19klQHDdrPY+Pc=
-----END EC PRIVATE KEY-----
```

- **Command to generate root csr-**
  openssl req -new -key rootkey_new.pem -out root.csr
  Output:

```
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]:IN
State or Province Name (full name) [Some-State]:Telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IITH
Organizational Unit Name (eg, section) []:CSE
Common Name (e.g. server FQDN or YOUR name) []:root
Email Address []:root@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:root
An optional company name []:IITH
```

- **Command to self sign the root csr to generate the root certificate-**
  openssl x509 -req -days 365 -in root.csr -signkey rootkey_ecc.pem -out root.crt
  Output:

```
Signature ok
subject=C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = CSE, CN =
root, emailAddress = root@gmail.com
Getting Private key
```

1. **Certificate of Alice in the alice1 LXD**

Get into Alice LXD by typing "lxc exec Alice1 bash"

a. Generating private key for Alice

Command - openssl genpkey -out alice_pvtkey_new.pem -algorithm RSA -pkeyopt rsa_keygen_bits:2048

Private key is as follows :

Command - openssl pkey -in alice_pvtkey_new.pem -text

Output

-----BEGIN PRIVATE KEY-----
MIIEvgIBADANBgkqhkiG9w0BAQEFAASCBKgwggSkAgEAAoIBAQDRaqGNNpafRbUB
vyCr0LwhFNsg5GecS8fDkl9UE9lwclZTcnqS2+qLJT5gwApWSk7meI26+wXB2ur6
CkRhAdKAR9Wivwd5YgfvLC8bw5ANihHVtR0A9W8xd3Sr0VepBSiYrfR5YkkMe4bo
eI+Bm3y60Tpq79RDElu9JpvMle7snFsqRggmh26ZIjWx+n/Qv9+4BP5ehg/9bwcn
HrwsEIJFBfiKKXfxWQz3XFsvpWP5ErhsaIKv7/yIOlJP3ST7qkAmBkt6Yxzmuj7A
N/JoJio4SVc/ao0ipzN98MuwEamH2/pfqL5OppFE+UgO1PYURa5ghk5k6CueRV2T
Ic3PlyDHAgMBAAECggEAVvC18TJghe8inn8CyVH3SKSn9Jfei/zKCl+SRA2ZkB1n
yW0VhjLWxL3Jj9EqeXiQGQ6jiKT+VuTT+MqaaMug4LDx+IbmS3ZvZNQrGpOuEnDn
xplSWTuL1jWJsmtOwVTq7bgKbvpm6U+/SnQC4AP1XHnQjuTScqYEgBqvQ9hbQM9I
bHf6HcUSK4HLFGygp9sfT1XJTIxEWMR2aud5RN3ubzfaA2FG7cvgSF93uVZfGezw
qlqlLUPG61nVdcM+wtZ2tzScXc+Myc6pDIblaBHcuRhKH8Mk6/kD9tmXrWj3GKJt
qzHLWA844LcIhyIfVvV6dOXK/2CsHNQ3qUfo+w/RAQKBgQDzVPTDdLppBN+HAq1Q
o6KUygLGcc5T5GM5JTmdLJns78rrfQQWPjxZJ+lLif77RQoqGb/YjGBvtP4lSsbX
Riodmna0Ug06p136dynkzm3RPI2+lEjBYTQXIriEKdlsjBTtON7GbU3ENu9nVJCo
Q3fjB8tPTZToLpaeM10f60qwsQKBgQDcUam56glj9vri3XIrmKR9MDM27JcO7MdQ
nDjvT1WVSNOw41bQSGziluwsPitCiZ4Fj3KMaEZr8II/FsT3napZEGJ4DCWcKBGl
NKgd3nNHMPprDZ0MHxt+qa0diCuzXwwEN23ZYK7T+v7gXlW0xwxoRC3JhiqVblMZ
bER4RSqG9wKBgQDFDeq9kn0XaRPf0QIsCKg0GDgRHr65X34fHA9BQopQv4w0EmTJ
4pKanOCoBNPSBoIJwBqFsoCyVf9ukdTfJvoaSSmSM9R0yG8p03DA049JX+FmEc2t
lRtLFRoaJY3nArY7uc9nH8NXgCl2I5yWNzUSTRCPY0OQl5HwAaaLn883sQKBgQCJ
CPYzoAElP0p/b1ODEFdHUoKYaAHxE49JcVHsb4EA8KJTBKRLurCXI15Uinqn0ScL
uFR8RqN+ocKxcP0kMtnwucM1fgCqpzWKdz0WDvXO0wT/LyF+l56nYuw9USQ1VioI
OV3G9l7nlNeRQLu4TJJIDBgan6u+9DcVugTAdezkswKBgAke/QiAbiCo0XqLGvtI
kXP7D3d72wkFFUBlh7idshXtwIGMmXA1t6hf5T8QKKS4zLRhU4o0cSkyS3jgvzg6
d3/ISslHbH7/94GTq6LFa0Hgsl5SQ/Id4moCJ8BJAruiXfOdcML5XYsbTLPfWldC
Fr1on2I5kPGdrshecE0l8e4e
-----END PRIVATE KEY-----

RSA Private-Key: (2048 bit, 2 primes)

modulus:

00:d1:6a:a1:8d:36:96:9f:45:b5:01:bf:20:ab:d0:
bc:21:14:db:20:e4:67:9c:4b:c7:c3:92:5f:54:13:
d9:70:72:56:53:72:7a:92:db:ea:8b:25:3e:60:c0:
0a:56:4a:4e:e6:78:8d:ba:fb:05:c1:da:ea:fa:0a:

44:61:01:d2:80:47:d5:a2:bf:07:79:62:07:ef:2c:
2f:1b:c3:90:0d:8a:11:d5:b5:1d:00:f5:6f:31:77:
74:ab:d1:57:a9:05:28:98:ad:f4:79:62:49:0c:7b:
86:e8:78:8f:81:9b:7c:ba:d1:3a:6a:ef:d4:43:12:
5b:bd:26:9b:cc:95:ee:ec:9c:5b:2a:46:08:26:87:
6e:99:22:35:b1:fa:7f:d0:bf:df:b8:04:fe:5e:86:
0f:fd:6f:07:27:1e:bc:2c:10:82:45:05:f8:8a:29:
77:f1:59:0c:f7:5c:5b:2f:a5:63:f9:12:b8:6c:68:
82:af:ef:fc:88:3a:52:4f:dd:24:fb:aa:40:26:06:
4b:7a:63:1c:e6:ba:3e:c0:37:f2:68:26:2a:38:49:
57:3f:6a:8d:22:a7:33:7d:f0:cb:b0:11:a9:87:db:
fa:5f:a8:be:4e:a6:91:44:f9:48:0e:d4:f6:14:45:
ae:60:86:4e:64:e8:2b:9e:45:5d:93:21:cd:cf:97:
20:c7
publicExponent: 65537 (0x10001)
privateExponent:
56:f0:b5:f1:32:60:85:ef:22:9e:7f:02:c9:51:f7:
48:a4:a7:f4:97:de:8b:fc:ca:0a:5f:92:44:0d:99:
90:1d:67:c9:6d:15:86:32:d6:c4:bd:c9:8f:d1:2a:
79:78:90:19:0e:a3:88:a4:fe:56:e4:d3:f8:ca:9a:
68:cb:a0:e0:b0:f1:f8:86:e6:4b:76:6f:64:d4:2b:
1a:93:ae:12:70:e7:c6:99:52:59:3b:8b:d6:35:89:
b2:6b:4e:c1:54:ea:ed:b8:0a:6e:fa:66:e9:4f:bf:
4a:74:02:e0:03:f5:5c:79:d0:8e:e4:d2:72:a6:04:
80:1a:af:43:d8:5b:40:cf:48:6c:77:fa:1d:c5:12:
2b:81:cb:14:6c:a0:a7:db:1f:4f:55:c9:4c:8c:44:
58:c4:76:6a:e7:79:44:dd:ee:6f:37:da:03:61:46:
ed:cb:e0:48:5f:77:b9:56:5f:19:ec:f0:aa:5a:a5:
2d:43:c6:eb:59:d5:75:c3:3e:c2:d6:76:b7:34:9c:
5d:cf:8c:c9:ce:a9:0c:86:e5:68:11:dc:b9:18:4a:
1f:c3:24:eb:f9:03:f6:d9:97:ad:68:f7:18:a2:6d:
ab:31:cb:58:0f:38:e0:b7:08:87:22:1f:56:f5:7a:
74:e5:ca:ff:60:ac:1c:d4:37:a9:47:e8:fb:0f:d1:
01
prime1:
00:f3:54:f4:c3:74:ba:69:04:df:87:02:ad:50:a3:
a2:94:ca:02:c6:71:ce:53:e4:63:39:25:39:9d:2c:
99:ec:ef:ca:eb:7d:04:16:3e:3c:59:27:e9:4b:89:
fe:fb:45:0a:2a:19:bf:d8:8c:60:6f:b4:fe:25:4a:
c6:d7:46:2a:1d:9a:76:b4:52:0d:3a:a7:5d:fa:77:
29:e4:ce:6d:d1:3c:8d:be:94:48:c1:61:34:17:22:
b8:84:29:d9:6c:8c:14:ed:38:de:c6:6d:4d:c4:36:
ef:67:54:90:a8:43:77:e3:07:cb:4f:4d:94:e8:2e:
96:9e:33:5d:1f:eb:4a:b0:b1
prime2:
00:dc:51:a9:b9:ea:09:63:f6:fa:e2:dd:72:2b:98:

```
    a4:7d:30:33:36:ec:97:0e:ec:c7:50:9c:38:ef:4f:
    55:95:48:d3:b0:e3:56:d0:48:6c:e2:96:ec:2c:3e:
    2b:42:89:9e:05:8f:72:8c:68:46:6b:f0:82:3f:16:
    c4:f7:9d:aa:59:10:62:78:0c:25:9c:28:11:a5:34:
    a8:1d:de:73:47:30:fa:6b:0d:9d:0c:1f:1b:7e:a9:
    ad:1d:88:2b:b3:5f:0c:04:37:6d:d9:60:ae:d3:fa:
    fe:e0:5e:55:b4:c7:0c:68:44:2d:c9:86:2a:95:6e:
    53:19:6c:44:78:45:2a:86:f7
exponent1:
    00:c5:0d:ea:bd:92:7d:17:69:13:df:d1:02:2c:08:
    a8:34:18:38:11:1e:be:b9:5f:7e:1f:1c:0f:41:42:
    8a:50:bf:8c:34:12:64:c9:e2:92:9a:9c:e0:a8:04:
    d3:d2:06:82:09:c0:1a:85:b2:80:b2:55:ff:6e:91:
    d4:df:26:fa:1a:49:29:92:33:d4:74:c8:6f:29:d3:
    70:c0:d3:8f:49:5f:e1:66:11:cd:ad:95:1b:4b:15:
    1a:1a:25:8d:e7:02:b6:3b:b9:cf:67:1f:c3:57:80:
    29:76:23:9c:96:37:35:12:4d:10:8f:63:43:90:97:
    91:f0:01:a6:8b:9f:cf:37:b1
exponent2:
    00:89:08:f6:33:a0:01:25:3f:4a:7f:6f:53:83:10:
    57:47:52:82:98:68:01:f1:13:8f:49:71:51:ec:6f:
    81:00:f0:a2:53:04:a4:4b:ba:b0:97:23:5e:54:8a:
    7a:a7:d1:27:0b:b8:54:7c:46:a3:7e:a1:c2:b1:70:
    fd:24:32:d9:f0:b9:c3:35:7e:00:aa:a7:35:8a:77:
    3d:16:0e:f5:ce:d3:04:ff:2f:21:7e:97:9e:a7:62:
    ec:3d:51:24:35:56:2a:08:39:5d:c6:f6:5e:e7:94:
    d7:91:40:bb:b8:4c:92:48:0c:18:1a:9f:ab:be:f4:
    37:15:ba:04:c0:75:ec:e4:b3
coefficient:
    09:1e:fd:08:80:6e:20:a8:d1:7a:8b:1a:fb:48:91:
    73:fb:0f:77:7b:db:09:05:15:40:65:87:b8:9d:b2:
    15:ed:c0:81:8c:99:70:35:b7:a8:5f:e5:3f:10:28:
    a4:b8:cc:b4:61:53:8a:34:71:29:32:4b:78:e0:bf:
    38:3a:77:7f:c8:4a:c9:47:6c:7e:ff:f7:81:93:ab:
    a2:c5:6b:41:e0:b2:5e:52:43:f2:1d:e2:6a:02:27:
    c0:49:02:bb:a2:5d:f3:9d:70:c2:f9:5d:8b:1b:4c:
    b3:df:5a:57:42:16:bd:68:9f:62:39:90:f1:9d:ae:
    c8:5e:70:4d:25:f1:ee:1e
```

b. Generating public key

Command - openssl pkey -in alice_pvtkey_new.pem -pubout -out alice_pbkey_new.pem

Public key is as follows :

Command - cat alice_pbkey_new.pem

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA0WqhjTaWn0W1Ab8gq9C8
IRTbIORnnEvHw5JfVBPZcHJWU3J6ktvqiyU+YMAKVkpO5niNuvsFwdrq+gpEYQHS
gEfVor8HeWIH7ywvG8OQDYoR1bUdAPVvMXd0q9FXqQUomK30eWJJDHuG6HiPgZt8
utE6au/UQxJbvSabzJXu7JxbKkYIJodumSI1sfp/0L/fuAT+XoYP/W8HJx68LBCC
RQX4iil38VkM91xbL6Vj+RK4bGiCr+/8iDpST90k+6pAJgZLemMc5ro+wDfyaCYq
OElXP2qNIqczffDLsBGph9v6X6i+TqaRRPlIDtT2FEWuYIZOZOgrnkVdkyHNz5cg
xwIDAQAB
-----END PUBLIC KEY-----
```

c. Generating Certificate signing request (CSR)

Command - openssl req -new -key alice_pvtkey_new.pem -out alice.csr

Output

```
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]:IN
State or Province Name (full name) [Some-State]:telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IITH
Organizational Unit Name (eg, section) []:CSE
Common Name (e.g. server FQDN or YOUR name) []:alice1
Email Address []:alice@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:alice1
An optional company name []:IITH


The generated csr is as follows :
```

Command - openssl req -text -in alice.csr

```
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = telangana, L = Hyderabad, O = IITH, OU = CSE, CN =
alice1, emailAddress = alice@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d1:6a:a1:8d:36:96:9f:45:b5:01:bf:20:ab:d0:
                    bc:21:14:db:20:e4:67:9c:4b:c7:c3:92:5f:54:13:
                    d9:70:72:56:53:72:7a:92:db:ea:8b:25:3e:60:c0:
                    0a:56:4a:4e:e6:78:8d:ba:fb:05:c1:da:ea:fa:0a:
                    44:61:01:d2:80:47:d5:a2:bf:07:79:62:07:ef:2c:
                    2f:1b:c3:90:0d:8a:11:d5:b5:1d:00:f5:6f:31:77:
                    74:ab:d1:57:a9:05:28:98:ad:f4:79:62:49:0c:7b:
                    86:e8:78:8f:81:9b:7c:ba:d1:3a:6a:ef:d4:43:12:
                    5b:bd:26:9b:cc:95:ee:ec:9c:5b:2a:46:08:26:87:
                    6e:99:22:35:b1:fa:7f:d0:bf:df:b8:04:fe:5e:86:
                    0f:fd:6f:07:27:1e:bc:2c:10:82:45:05:f8:8a:29:
                    77:f1:59:0c:f7:5c:5b:2f:a5:63:f9:12:b8:6c:68:
                    82:af:ef:fc:88:3a:52:4f:dd:24:fb:aa:40:26:06:
                    4b:7a:63:1c:e6:ba:3e:c0:37:f2:68:26:2a:38:49:
                    57:3f:6a:8d:22:a7:33:7d:f0:cb:b0:11:a9:87:db:
                    fa:5f:a8:be:4e:a6:91:44:f9:48:0e:d4:f6:14:45:
                    ae:60:86:4e:64:e8:2b:9e:45:5d:93:21:cd:cf:97:
                    20:c7
                Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :IITH
            challengePassword        :alice1
    Signature Algorithm: sha256WithRSAEncryption
         96:5d:2f:ae:53:92:57:27:ae:c4:af:04:af:de:97:d7:65:d4:
         dd:bb:b9:ec:c7:52:d7:e4:09:e7:f4:7a:31:61:51:6c:7a:b9:
         fe:c9:c9:df:83:de:cc:38:99:a1:51:c6:03:a7:79:61:57:09:
         23:3c:db:bb:f8:21:c5:cb:d8:45:23:88:49:c9:d4:a6:0c:61:
         4d:95:ec:88:34:da:97:7b:72:29:a9:63:79:7d:46:a0:06:53:
         2f:2c:36:52:8c:aa:51:2d:3f:85:28:a5:9f:b3:85:4c:d5:d8:
         b1:c8:99:23:33:cf:b8:3c:29:f8:a9:6a:ad:dd:6e:ae:d8:3f:
         d6:67:ab:45:03:d1:2c:e7:c1:8a:fc:e6:af:fc:8b:59:86:b8:
         a6:8a:f8:8b:fa:5c:50:af:29:d7:22:53:22:29:82:8e:d8:c5:
         35:ec:81:82:d3:33:00:53:86:30:13:4b:a9:ec:cb:4a:07:7b:
         73:e2:e4:5a:0c:e0:30:e2:85:ec:c3:1b:83:a7:d8:16:01:2c:
```

```
        16:81:46:07:2d:e7:c9:6f:9a:b8:0b:04:1e:8f:ee:0e:56:21:
        de:67:8d:48:60:da:95:a7:e7:cd:c1:76:f4:6e:0e:95:50:6d:
        8c:6a:44:74:30:47:30:f9:37:f1:39:b7:e5:1b:39:45:c3:44:
        ba:80:c8:60
-----BEGIN CERTIFICATE REQUEST-----
MIIC9TCCAd0CAQAwgYMxCzAJBgNVBAYTAklOMRIwEAYDVQQIDAl0ZWxhbmdhbmEx
EjAQBgNVBAcMCUh5ZGVyYWJhZDENMAsGA1UECgwESUlUSDEMMAoGA1UECwwDQ1NF
MQ8wDQYDVQQDDAZhbGljZTExHjAcBgkqhkiG9w0BCQEWD2FsaWNlQGdtYWlsLmNv
bTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANFqoY02lp9FtQG/IKvQ
vCEU2yDkZ5xLx8OSX1QT2XByVlNyepLb6oslPmDAClZKTuZ4jbr7BcHa6voKRGEB
0oBH1aK/B3liB+8sLxvDkA2KEdW1HQD1bzF3dKvRV6kFKJit9HliSQx7huh4j4Gb
fLrROmrv1EMSW70mm8yV7uycWypGCCaHbpkiNbH6f9C/37gE/l6GD/1vBycevCwQ
gkUF+Iopd/FZDPdcWy+lY/kSuGxogq/v/Ig6Uk/dJPuqQCYGS3pjHOa6PsA38mgm
KjhJVz9qjSKnM33wy7ARqYfb+l+ovk6mkUT5SA7U9hRFrmCGTmToK55FXZMhzc+X
IMcCAwEAAaAsMBMGCSqGSIb3DQEJAjEGDARJSVRIMBUGCSqGSIb3DQEJBzEIDAZh
bGljZTEwDQYJKoZIhvcNAQELBQADggEBAJZdL65TklcnrsSvBK/el9dl1N27uezH
UtfkCef0ejFhUWx6uf7Jyd+D3sw4maFRxgOneWFXCSM827v4IcXL2EUjiEnJ1KYM
YU2V7Ig02pd7cimpY3l9RqAGUy8sNlKMqlEtP4UopZ+zhUzV2LHImSMzz7g8Kfip
aq3dbq7YP9Znq0UD0SznwYr85q/8i1mGuKaK+Iv6XFCvKdciUyIpgo7YxTXsgYLT
MwBThjATS6nsy0oHe3Pi5FoM4DDihezDG4On2BYBLBaBRgct58lvmrgLBB6P7g5W
Id5njUhg2pWn583BdvRuDpVQbYxqRHQwRzD5N/E5t+UbOUXDRLqAyGA=
-----END CERTIFICATE REQUEST-----
```

2. Certificate of Bob in bob1 LXD

Get into Bob LXD by typing "lxc exec bob1 bash"

a. Generating private key for Bob

Command - openssl genpkey -out bob_pvtkey_new.pem -algorithm RSA -pkeyopt rsa_keygen_bits:2048

Private key is as follows :
Command - openssl pkey -in bob_pvtkey_new.pem -text

```
-----BEGIN PRIVATE KEY-----
MIIEwAIBADANBgkqhkiG9w0BAQEFAASCBKowggSmAgEAAoIBAQCZN4t8hGZCRw4l
PCqgjxJHoXAD7oPBask2wpkpjkl2//T3S3Q9IxpSTuQFMQqnL7C/J56/kjWUrsCY
wgtIjTXMk58gR1oAk0gArLf3y0dU+i7rsxqm+BTp2CzTdFsTaju7niII5cJ7t/7j
2O1PaYvYj5jDy18/LveCjOMhrRk+k0LQ/Fe3+88USIU/57DnWOlzx6aEPu4+tigh
7U8meKA65QxoYPDpgZjscNttzQAg8+FEqog8d9E8202dOwV8hojM7ZAhf9sm9+yD
```

```
+TyXkAeQEQ1Ctg5S4RkwlgMSNaICH22R4WwRwpPk8r/wA7QqUo4p5ebgHwhjxgVO
AOXO8VQfAgMBAAECggEBAIih0t7whux1xvSHig/yZj/AhE7NRWuXWeZedQ1s2LCp
In/hJsBabF+YAJlsmW5sBfcBUvzNWsDDcfcHm95QJvjPcCjt2CIJNT18yPXNrkfm
by87Yfoh4yyB3b+X1JwksS3nn7xFgECp3UC8syuNGOu/amC4HLEQpiIil23xmkCE
2ucoSkAg2whZnDQtSnD75KZo+pswubCKyrfgQh6mLewx9ujZWRhn1qF8eEvwZLNE
5a+D6sLrlN1WecPvywinCdBOle/UVrPUYuRT+2TcTCa8IVkQuVoXdBXXMTnhnrpO
kd+ZNp2VqEU7/oDOsDP53zdPJBzAbIUEh1mBBlGGBxECgYEAxma/sx15bJ5D8WDV
EaLNsA3RRKOhnEajn3bkDE4j4d63c0E/mK0z6Du0or6yzBsy5LlkxS1k176ljg+B
Dga+uKa0HA9+MxiUrk5BOodVPGBslT50JMYnLVRV78pJbf2irtCfugeDXxwOed8z
h/tUbLuWp+yLHHDpM3Oiecc2vgkCgYEAxbKtlFQqP0Xab7VQFZV0uVnZGYbSEHkb
MnAx+fKo8J5QHEdvIYR5ZITvkZTE5dJiDlkpx3Y6wxpMIH9WxihOITJ5d3D+K/lC
y3EgHeIz0xRO1BhGeYuZ8s+HURrNLvDn56dDaWAhhpIKOv9X8yhNuHAU/6xxdLxj
dN8X9NADiucCgYEAku5PNAA1B5rh6GX1Bb2TJLnm7DVen1S1Ioiy9OX9ru344FNH
HPPQiEGHUit2ULAn7bnbBifLEffvtZiH0/Js1mUm580VghPHMJtltUm6dx9V6uU4
sG9LXkeh6RbfcKSxEVQRpnyf7xThJ6KEEEWyTWBTf4Mvtt1r5x3RCuzc/rkCgYEA
mtwDrBThov6MxMQ6Hy4baLVAogZBtjWnYDMScr/qJdFc30n1e0PR8zNOlbL7KStL
NAIT//JUxU9sImidCEu6J7bp/u27q6Zs8/+BMOdnwQg/V1RMoBkCVnjJfnaIEyUS
Axu6amiq2ka7TMuHkSkY+EiTqlLh7JdFMN8wXMw0ejECgYEAnntE73HfBm5i9Gm4
kfdMCYgJRApPxwbvHhq9T8bDW3tgw9u1O/MhrCyprLfYDWq9JkDoDWMpu+wo1xNw
t+ZyZGUHvtDPSEbLbdt8TzJ0hpExeRv9IZ3lVXSMkw5JKIKPB2/6Krqi92sJ8CSW
Y1B9buJ9qBVqPFzB+yu4ui7GWE4=
-----END PRIVATE KEY-----
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:99:37:8b:7c:84:66:42:47:0e:25:3c:2a:a0:8f:
    12:47:a1:70:03:ee:83:c1:6a:c9:36:c2:99:29:8e:
    49:76:ff:f4:f7:4b:74:3d:23:1a:52:4e:e4:05:31:
    0a:a7:2f:b0:bf:27:9e:bf:92:35:94:ae:c0:98:c2:
    0b:48:8d:35:cc:93:9f:20:47:5a:00:93:48:00:ac:
    b7:f7:cb:47:54:fa:2e:eb:b3:1a:a6:f8:14:e9:d8:
    2c:d3:74:5b:13:6a:3b:bb:9e:22:08:e5:c2:7b:b7:
    fe:e3:d8:ed:4f:69:8b:d8:8f:98:c3:cb:5f:3f:2e:
    f7:82:8c:e3:21:ad:19:3e:93:42:d0:fc:57:b7:fb:
    cf:14:48:85:3f:e7:b0:e7:58:e9:73:c7:a6:84:3e:
    ee:3e:b6:28:21:ed:4f:26:78:a0:3a:e5:0c:68:60:
    f0:e9:81:98:ec:70:db:6d:cd:00:20:f3:e1:44:aa:
    88:3c:77:d1:3c:db:4d:9d:3b:05:7c:86:88:cc:ed:
    90:21:7f:db:26:f7:ec:83:f9:3c:97:90:07:90:11:
    0d:42:b6:0e:52:e1:19:30:96:03:12:35:a2:02:1f:
    6d:91:e1:6c:11:c2:93:e4:f2:bf:f0:03:b4:2a:52:
    8e:29:e5:e6:e0:1f:08:63:c6:05:4e:00:e5:ce:f1:
    54:1f
publicExponent: 65537 (0x10001)
privateExponent:
    00:88:a1:d2:de:f0:86:ec:75:c6:f4:87:8a:0f:f2:
```

```
    66:3f:c0:84:4e:cd:45:6b:97:59:e6:5e:75:0d:6c:
    d8:b0:a9:22:7f:e1:26:c0:5a:6c:5f:98:00:99:6c:
    99:6e:6c:05:f7:01:52:fc:cd:5a:c0:c3:71:f7:07:
    9b:de:50:26:f8:cf:70:28:ed:d8:22:09:35:3d:7c:
    c8:f5:cd:ae:47:e6:6f:2f:3b:61:fa:21:e3:2c:81:
    dd:bf:97:d4:9c:24:b1:2d:e7:9f:bc:45:80:40:a9:
    dd:40:bc:b3:2b:8d:18:eb:bf:6a:60:b8:1c:b1:10:
    a6:22:22:97:6d:f1:9a:40:84:da:e7:28:4a:40:20:
    db:08:59:9c:34:2d:4a:70:fb:e4:a6:68:fa:9b:30:
    b9:b0:8a:ca:b7:e0:42:1e:a6:2d:ec:31:f6:e8:d9:
    59:18:67:d6:a1:7c:78:4b:f0:64:b3:44:e5:af:83:
    ea:c2:eb:94:dd:56:79:c3:ef:cb:08:a7:09:d0:4e:
    95:ef:d4:56:b3:d4:62:e4:53:fb:64:dc:4c:26:bc:
    21:59:10:b9:5a:17:74:15:d7:31:39:e1:9e:ba:4e:
    91:df:99:36:9d:95:a8:45:3b:fe:80:ce:b0:33:f9:
    df:37:4f:24:1c:c0:6c:85:04:87:59:81:06:51:86:
    07:11
prime1:
    00:c6:66:bf:b3:1d:79:6c:9e:43:f1:60:d5:11:a2:
    cd:b0:0d:d1:44:a3:a1:9c:46:a3:9f:76:e4:0c:4e:
    23:e1:de:b7:73:41:3f:98:ad:33:e8:3b:b4:a2:be:
    b2:cc:1b:32:e4:b9:64:c5:2d:64:d7:be:a5:8e:0f:
    81:0e:06:be:b8:a6:b4:1c:0f:7e:33:18:94:ae:4e:
    41:3a:87:55:3c:60:6c:95:3e:74:24:c6:27:2d:54:
    55:ef:ca:49:6d:fd:a2:ae:d0:9f:ba:07:83:5f:1c:
    0e:79:df:33:87:fb:54:6c:bb:96:a7:ec:8b:1c:70:
    e9:33:73:a2:79:c7:36:be:09
prime2:
    00:c5:b2:ad:94:54:2a:3f:45:da:6f:b5:50:15:95:
    74:b9:59:d9:19:86:d2:10:79:1b:32:70:31:f9:f2:
    a8:f0:9e:50:1c:47:6f:21:84:79:64:84:ef:91:94:
    c4:e5:d2:62:0e:59:29:c7:76:3a:c3:1a:4c:20:7f:
    56:c6:28:4e:21:32:79:77:70:fe:2b:f9:42:cb:71:
    20:1d:e2:33:d3:14:4e:d4:18:46:79:8b:99:f2:cf:
    87:51:1a:cd:2e:f0:e7:e7:a7:43:69:60:21:86:92:
    0a:3a:ff:57:f3:28:4d:b8:70:14:ff:ac:71:74:bc:
    63:74:df:17:f4:d0:03:8a:e7
exponent1:
    00:92:ee:4f:34:00:35:07:9a:e1:e8:65:f5:05:bd:
    93:24:b9:e6:ec:35:5e:9f:54:b5:22:88:b2:f4:e5:
    fd:ae:ed:f8:e0:53:47:1c:f3:d0:88:41:87:52:2b:
    76:50:b0:27:ed:b9:db:06:27:cb:11:f7:ef:b5:98:
    87:d3:f2:6c:d6:65:26:e7:cd:15:82:13:c7:30:9b:
    65:b5:49:ba:77:1f:55:ea:e5:38:b0:6f:4b:5e:47:
    a1:e9:16:df:70:a4:b1:11:54:11:a6:7c:9f:ef:14:
```

```
    e1:27:a2:84:10:45:b2:4d:60:53:7f:83:2f:b6:dd:
    6b:e7:1d:d1:0a:ec:dc:fe:b9
exponent2:
    00:9a:dc:03:ac:14:e1:a2:fe:8c:c4:c4:3a:1f:2e:
    1b:68:b5:40:a2:06:41:b6:35:a7:60:33:12:72:bf:
    ea:25:d1:5c:df:49:f5:7b:43:d1:f3:33:4e:95:b2:
    fb:29:2b:4b:34:02:13:ff:f2:54:c5:4f:6c:22:68:
    9d:08:4b:ba:27:b6:e9:fe:ed:bb:ab:a6:6c:f3:ff:
    81:30:e7:67:c1:08:3f:57:54:4c:a0:19:02:56:78:
    c9:7e:76:88:13:25:12:03:1b:ba:6a:68:aa:da:46:
    bb:4c:cb:87:91:29:18:f8:48:93:aa:52:e1:ec:97:
    45:30:df:30:5c:cc:34:7a:31
coefficient:
    00:9e:7b:44:ef:71:df:06:6e:62:f4:69:b8:91:f7:
    4c:09:88:09:44:0a:4f:c7:06:ef:1e:1a:bd:4f:c6:
    c3:5b:7b:60:c3:db:b5:3b:f3:21:ac:2c:a9:ac:b7:
    d8:0d:6a:bd:26:40:e8:0d:63:29:bb:ec:28:d7:13:
    70:b7:e6:72:64:65:07:be:d0:cf:48:46:cb:6d:db:
    7c:4f:32:74:86:91:31:79:1b:fd:21:9d:e5:55:74:
    8c:93:0e:49:28:82:8f:07:6f:fa:2a:ba:a2:f7:6b:
    09:f0:24:96:63:50:7d:6e:e2:7d:a8:15:6a:3c:5c:
    c1:fb:2b:b8:ba:2e:c6:58:4e
```

b. Generating public key

Command - openssl pkey -in bob_pvtkey_new.pem -pubout -out bob_pbkey_new.pem

Public key is as follows :
Command - cat bob_pbkey_new.pem

```
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAmTeLfIRmQkcOJTwqoI8S
R6FwA+6DwWrJNsKZKY5Jdv/090t0PSMaUk7kBTEKpy+wvyeev5I1lK7AmMILSI01
zJOfIEdaAJNIAKy398tHVPou67MapvgU6dgs03RbE2o7u54iCOXCe7f+49jtT2mL
2I+Yw8tfPy73gozjIa0ZPpNC0PxXt/vPFEiFP+ew51jpc8emhD7uPrYoIe1PJnig
OuUMaGDw6YGY7HDbbc0AIPPhRKqIPHfRPNtNnTsFfIaIzO2QIX/bJvfsg/k8l5AH
kBENQrYOUuEZMJYDEjWiAh9tkeFsEcKT5PK/8AO0KlKOKeXm4B8IY8YFTgDlzvFU
HwIDAQAB
-----END PUBLIC KEY-----
```

c. Generating Certificate signing request (CSR)

Command - openssl req -new -key bob_pvtkey_new.pem -out bob.csr

Output:

```
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]:IN
State or Province Name (full name) [Some-State]:Telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IITH
Organizational Unit Name (eg, section) []:CSE
Common Name (e.g. server FQDN or YOUR name) []:bob1
Email Address []:bob@gmail.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:bob1
An optional company name []:IITH
```

Command - openssl req -text -in bob.csr

```
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = CSE, CN =
bob1, emailAddress = bob@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:99:37:8b:7c:84:66:42:47:0e:25:3c:2a:a0:8f:
                    12:47:a1:70:03:ee:83:c1:6a:c9:36:c2:99:29:8e:
                    49:76:ff:f4:f7:4b:74:3d:23:1a:52:4e:e4:05:31:
```

0a:a7:2f:b0:bf:27:9e:bf:92:35:94:ae:c0:98:c2:
0b:48:8d:35:cc:93:9f:20:47:5a:00:93:48:00:ac:
b7:f7:cb:47:54:fa:2e:eb:b3:1a:a6:f8:14:e9:d8:
2c:d3:74:5b:13:6a:3b:bb:9e:22:08:e5:c2:7b:b7:
fe:e3:d8:ed:4f:69:8b:d8:8f:98:c3:cb:5f:3f:2e:
f7:82:8c:e3:21:ad:19:3e:93:42:d0:fc:57:b7:fb:
cf:14:48:85:3f:e7:b0:e7:58:e9:73:c7:a6:84:3e:
ee:3e:b6:28:21:ed:4f:26:78:a0:3a:e5:0c:68:60:
f0:e9:81:98:ec:70:db:6d:cd:00:20:f3:e1:44:aa:
88:3c:77:d1:3c:db:4d:9d:3b:05:7c:86:88:cc:ed:
90:21:7f:db:26:f7:ec:83:f9:3c:97:90:07:90:11:
0d:42:b6:0e:52:e1:19:30:96:03:12:35:a2:02:1f:
6d:91:e1:6c:11:c2:93:e4:f2:bf:f0:03:b4:2a:52:
8e:29:e5:e6:e0:1f:08:63:c6:05:4e:00:e5:ce:f1:
54:1f
Exponent: 65537 (0x10001)
Attributes:
unstructuredName :IITH
challengePassword :bob1
Signature Algorithm: sha256WithRSAEncryption
24:ae:19:b1:3a:8e:00:2c:1e:2b:90:f7:d4:9c:b1:49:2b:97:
fc:d5:33:5e:0c:cb:db:50:9a:ab:12:31:4b:ec:38:a4:82:dd:
5d:94:33:62:1d:43:e2:db:36:8b:8b:f8:03:1d:2b:d3:b2:17:
d3:02:cd:1a:57:df:5b:7e:04:f0:a8:40:71:3f:d0:f6:4b:75:
3c:4d:60:6c:54:dc:f3:0f:0f:63:b0:fc:3b:52:43:48:d8:ce:
bf:5c:32:21:9f:ca:97:9c:40:70:bc:87:b7:b1:6e:41:c4:60:
7c:eb:3a:ee:dc:8a:98:00:bc:0e:a0:ee:91:bb:e6:55:a5:c2:
e5:a1:4d:19:5a:a9:be:38:a3:10:3f:16:7e:e1:f1:17:82:eb:
6d:4e:02:0e:1f:bb:2f:d4:62:80:bf:b7:2f:7c:c2:79:83:a3:
b0:84:b5:f7:79:16:43:00:31:a8:54:c8:5d:28:8a:c5:35:2f:
35:9b:90:f5:b1:31:61:ea:44:ec:79:2e:65:3b:d2:b4:3a:9b:
b0:26:83:4b:a2:41:71:07:84:d6:2e:31:d7:69:81:70:f1:c1:
a2:3d:02:41:82:ed:c1:38:1c:28:cd:03:20:c9:75:18:c4:b7:
33:04:34:c5:27:a8:64:5c:da:3e:b7:3b:7c:a2:39:a7:7e:f2:
6b:5b:75:d8
-----BEGIN CERTIFICATE REQUEST-----
MIIC7jCCAdYCAQAwfzELMAkGA1UEBhMCSU4xEjAQBgNVBAgMCVRlbGFuZ
2FuYTES
MBAGA1UEBwwJSHlkZXJhYmFkMQ0wCwYDVQQKDARJSVRIMQwwCgYDV
QQLDANDU0Ux

DTALBgNVBAMMBGJvYjExHDAaBgkqhkiG9w0BCQEWDWJvYkBnbWFpbC5jb
20wggEi
MA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCZN4t8hGZCRw4lPCqgjxJ
HoXAD
7oPBask2wpkpjkl2//T3S3Q9IxpSTuQFMQqnL7C/J56/kjWUrsCYwgtIjTXMk58g
R1oAk0gArLf3y0dU+i7rsxqm+BTp2CzTdFsTaju7niII5cJ7t/7j2O1PaYvYj5jD
y18/LveCjOMhrRk+k0LQ/Fe3+88USIU/57DnWOlzx6aEPu4+tigh7U8meKA65Qxo
YPDpgZjscNttzQAg8+FEqog8d9E8202dOwV8hojM7ZAhf9sm9+yD+TyXkAeQEQ1
C
tg5S4RkwlgMSNaICH22R4WwRwpPk8r/wA7QqUo4p5ebgHwhjxgVOAOXO8VQfA
gMB
AAGgKjATBgkqhkiG9w0BCQIxBgwESUlUSDATBgkqhkiG9w0BCQcxBgwEYm9i
MTAN
BgkqhkiG9w0BAQsFAAOCAQEAJK4ZsTqOACweK5D31JyxSSuX/NUzXgzL21Caq
xIx
S+w4pILdXZQzYh1D4ts2i4v4Ax0r07IX0wLNGlffW34E8KhAcT/Q9kt1PE1gbFTc
8w8PY7D8O1JDSNjOv1wyIZ/Kl5xAcLyHt7FuQcRgfOs67tyKmAC8DqDukbvmVaX
C
5aFNGVqpvjijED8WfuHxF4LrbU4CDh+7L9RigL+3L3zCeYOjsIS193kWQwAxqFTI
XSiKxTUvNZuQ9bExYepE7HkuZTvStDqbsCaDS6JBcQeE1i4x12mBcPHBoj0CQYL
t
wTgcKM0DIMl1GMS3MwQ0xSeoZFzaPrc7fKI5p37ya1t12A==
-----END CERTIFICATE REQUEST-----

3. Generating Certificate signing request (CSR)

a. Verifying alice.csr

Command - openssl req -text -noout -verify -in alice.csr

Output

verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = telangana, L = Hyderabad, O = IITH, OU = CSE, CN = alice1, emailAddress = alice@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:d1:6a:a1:8d:36:96:9f:45:b5:01:bf:20:ab:d0:

```
            bc:21:14:db:20:e4:67:9c:4b:c7:c3:92:5f:54:13:
            d9:70:72:56:53:72:7a:92:db:ea:8b:25:3e:60:c0:
            0a:56:4a:4e:e6:78:8d:ba:fb:05:c1:da:ea:fa:0a:
            44:61:01:d2:80:47:d5:a2:bf:07:79:62:07:ef:2c:
            2f:1b:c3:90:0d:8a:11:d5:b5:1d:00:f5:6f:31:77:
            74:ab:d1:57:a9:05:28:98:ad:f4:79:62:49:0c:7b:
            86:e8:78:8f:81:9b:7c:ba:d1:3a:6a:ef:d4:43:12:
            5b:bd:26:9b:cc:95:ee:ec:9c:5b:2a:46:08:26:87:
            6e:99:22:35:b1:fa:7f:d0:bf:df:b8:04:fe:5e:86:
            0f:fd:6f:07:27:1e:bc:2c:10:82:45:05:f8:8a:29:
            77:f1:59:0c:f7:5c:5b:2f:a5:63:f9:12:b8:6c:68:
            82:af:ef:fc:88:3a:52:4f:dd:24:fb:aa:40:26:06:
            4b:7a:63:1c:e6:ba:3e:c0:37:f2:68:26:2a:38:49:
            57:3f:6a:8d:22:a7:33:7d:f0:cb:b0:11:a9:87:db:
            fa:5f:a8:be:4e:a6:91:44:f9:48:0e:d4:f6:14:45:
            ae:60:86:4e:64:e8:2b:9e:45:5d:93:21:cd:cf:97:
            20:c7
        Exponent: 65537 (0x10001)
    Attributes:
        unstructuredName        :IITH
        challengePassword       :alice1
Signature Algorithm: sha256WithRSAEncryption
     96:5d:2f:ae:53:92:57:27:ae:c4:af:04:af:de:97:d7:65:d4:
     dd:bb:b9:ec:c7:52:d7:e4:09:e7:f4:7a:31:61:51:6c:7a:b9:
     fe:c9:c9:df:83:de:cc:38:99:a1:51:c6:03:a7:79:61:57:09:
     23:3c:db:bb:f8:21:c5:cb:d8:45:23:88:49:c9:d4:a6:0c:61:
     4d:95:ec:88:34:da:97:7b:72:29:a9:63:79:7d:46:a0:06:53:
     2f:2c:36:52:8c:aa:51:2d:3f:85:28:a5:9f:b3:85:4c:d5:d8:
     b1:c8:99:23:33:cf:b8:3c:29:f8:a9:6a:ad:dd:6e:ae:d8:3f:
     d6:67:ab:45:03:d1:2c:e7:c1:8a:fc:e6:af:fc:8b:59:86:b8:
     a6:8a:f8:8b:fa:5c:50:af:29:d7:22:53:22:29:82:8e:d8:c5:
     35:ec:81:82:d3:33:00:53:86:30:13:4b:a9:ec:cb:4a:07:7b:
     73:e2:e4:5a:0c:e0:30:e2:85:ec:c3:1b:83:a7:d8:16:01:2c:
     16:81:46:07:2d:e7:c9:6f:9a:b8:0b:04:1e:8f:ee:0e:56:21:
     de:67:8d:48:60:da:95:a7:e7:cd:c1:76:f4:6e:0e:95:50:6d:
     8c:6a:44:74:30:47:30:f9:37:f1:39:b7:e5:1b:39:45:c3:44:
     ba:80:c8:60
```

b. Signing alice.csr to generate alice.crt

Command - openssl x509 -req -in alice.csr -CA root.crt -CAkey rootkey_ecc.pem -CAcreateserial -out alice.crt -days 200 -sha256

Output :

Signature ok
subject=C = IN, ST = telangana, L = Hyderabad, O = IITH, OU = CSE, CN = alice1, emailAddress = alice@gmail.com
Getting CA Private Key

c. Verifying bob.csr

Command: openssl req -text -noout -verify -in bob.csr

Output:

```
verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = CSE, CN =
bob1, emailAddress = bob@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:99:37:8b:7c:84:66:42:47:0e:25:3c:2a:a0:8f:
                    12:47:a1:70:03:ee:83:c1:6a:c9:36:c2:99:29:8e:
                    49:76:ff:f4:f7:4b:74:3d:23:1a:52:4e:e4:05:31:
                    0a:a7:2f:b0:bf:27:9e:bf:92:35:94:ae:c0:98:c2:
                    0b:48:8d:35:cc:93:9f:20:47:5a:00:93:48:00:ac:
                    b7:f7:cb:47:54:fa:2e:eb:b3:1a:a6:f8:14:e9:d8:
                    2c:d3:74:5b:13:6a:3b:bb:9e:22:08:e5:c2:7b:b7:
                    fe:e3:d8:ed:4f:69:8b:d8:8f:98:c3:cb:5f:3f:2e:
                    f7:82:8c:e3:21:ad:19:3e:93:42:d0:fc:57:b7:fb:
                    cf:14:48:85:3f:e7:b0:e7:58:e9:73:c7:a6:84:3e:
                    ee:3e:b6:28:21:ed:4f:26:78:a0:3a:e5:0c:68:60:
                    f0:e9:81:98:ec:70:db:6d:cd:00:20:f3:e1:44:aa:
                    88:3c:77:d1:3c:db:4d:9d:3b:05:7c:86:88:cc:ed:
                    90:21:7f:db:26:f7:ec:83:f9:3c:97:90:07:90:11:
                    0d:42:b6:0e:52:e1:19:30:96:03:12:35:a2:02:1f:
                    6d:91:e1:6c:11:c2:93:e4:f2:bf:f0:03:b4:2a:52:
                    8e:29:e5:e6:e0:1f:08:63:c6:05:4e:00:e5:ce:f1:
                    54:1f
                Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :IITH
            challengePassword        :bob1
    Signature Algorithm: sha256WithRSAEncryption
```

```
        24:ae:19:b1:3a:8e:00:2c:1e:2b:90:f7:d4:9c:b1:49:2b:97:
        fc:d5:33:5e:0c:cb:db:50:9a:ab:12:31:4b:ec:38:a4:82:dd:


5d:94:33:62:1d:43:e2:db:36:8b:8b:f8:03:1d:2b:d3:b2:17:
        d3:02:cd:1a:57:df:5b:7e:04:f0:a8:40:71:3f:d0:f6:4b:75:
        3c:4d:60:6c:54:dc:f3:0f:0f:63:b0:fc:3b:52:43:48:d8:ce:
        bf:5c:32:21:9f:ca:97:9c:40:70:bc:87:b7:b1:6e:41:c4:60:
        7c:eb:3a:ee:dc:8a:98:00:bc:0e:a0:ee:91:bb:e6:55:a5:c2:
        e5:a1:4d:19:5a:a9:be:38:a3:10:3f:16:7e:e1:f1:17:82:eb:
        6d:4e:02:0e:1f:bb:2f:d4:62:80:bf:b7:2f:7c:c2:79:83:a3:
        b0:84:b5:f7:79:16:43:00:31:a8:54:c8:5d:28:8a:c5:35:2f:
        35:9b:90:f5:b1:31:61:ea:44:ec:79:2e:65:3b:d2:b4:3a:9b:
        b0:26:83:4b:a2:41:71:07:84:d6:2e:31:d7:69:81:70:f1:c1:
        a2:3d:02:41:82:ed:c1:38:1c:28:cd:03:20:c9:75:18:c4:b7:
        33:04:34:c5:27:a8:64:5c:da:3e:b7:3b:7c:a2:39:a7:7e:f2:
        6b:5b:75:d8
```

d. Signing bob.csr

Command - openssl x509 -req -in bob.csr -CA root.crt -CAkey rootkey.pem -CAcreateserial -out bob.crt -days 200 -sha256

Output :

```
Signature ok
subject=C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = CSE, CN = bob1,
emailAddress = bob@gmail.com
Getting CA Private Key
```

e. Verifying alice.crt

Command - openssl verify -trusted root.crt alice.crt

Output - alice.crt: OK

f. Verifying bob.crt

Command - openssl verify -trusted root.crt bob.crt

Output - bob.crt: OK

Digital Signature and Verification:

Signing of Alice's CSR and its verification:

1. **Creation of SHA 1 digest:**
   Command: openssl dgst -sha1 -out alice_csr_digest alice.csr
2. **Signing the Alice's CSR with its private key**
   Command: openssl pkeyutl -sign -in alice_csr_digest -out alice_csr_sign -inkey alice_pvtkey_new.pem
3. **Verifying the signature:**
   Command: openssl pkeyutl -verify -sigfile alice_csr_sign -in alice_csr_digest -inkey alice_pbkey_new.pem -pubin

```
root@alice1:~# openssl dgst -sha1 -out alice_csr_digest alice.csr
root@alice1:~# openssl pkeyutl -sign -in alice_csr_digest -out alice_csr_sign -inkey alice_pvtkey_new.pem
root@alice1:~# openssl pkeyutl -verify -sigfile alice_csr_sign -in alice_csr_digest -inkey alice_pbkey_new.pem -pubin
Signature Verified Successfully
root@alice1:~#
```

Verification of alice.crt using root certificate

Command: openssl verify -verbose -CAfile root.crt alice.crt

```
ns@ns08:~$ lxc exec alice1 bash
root@alice1:~# openssl verify -verbose -CAfile root.crt alice.crt
alice.crt: OK
```

Signing of Bob's CSR and its verification:

1. **Creation of SHA 1 digest:**
   Command: openssl dgst -sha1 -out bob_csr_digest bob.csr
2. **Signing the Alice's CSR with its private key**
   Command: openssl pkeyutl -sign -in bob_csr_digest -out bob_csr_sign -inkey bob_pvtkey_new.pem
3. **Verifying the signature:**
   Command: openssl pkeyutl -verify -sigfile bob_csr_sign -in bob_csr_digest -inkey bob_pbkey_new.pem -pubin

```
root@bob1:~# openssl dgst -sha1 -out bob_csr_digest bob.csr
root@bob1:~# openssl pkeyutl -sign -in bob_csr_digest -out bob_csr_sign -inkey bob_pvtkey_new.pem
root@bob1:~# openssl pkeyutl -verify -sigfile bob_csr_sign -in bob_csr_digest -inkey bob_pbkey_new.pem -pubin
Signature Verified Successfully
```

Verification of bob.crt using root certificate

Command: openssl verify -verbose -CAfile root.crt bob.crt

```
root@bob1:~# openssl verify -verbose -CAfile root.crt bob.crt
bob.crt: OK
```

Root Certificate Verification

Signing of Root's Certificate and its verification:

1. **Creation of SHA 1 digest:**
   Command: openssl dgst -sha1 -out root_digest root.crt
2. **Signing the Alice's CSR with its private key**
   Command: openssl pkeyutl -sign -in root_digest -out bob_csr_sign -inkey rootkey_ecc.pem
3. **Verifying the signature:**
   Command: openssl pkeyutl -verify -sigfile bob_csr_sign -in bob_csr_digest -inkey bob_pbkey_new.pem -pubin

```
ns@ns08:~$ openssl pkey -in rootkey_ecc.pem -pubout -out rootkey_pbkey_ecc.pem
ns@ns08:~$ openssl dgst -sha1 -out root_digest root.crt
ns@ns08:~$ openssl pkeyutl -sign -in root_digest -out root_sign -inkey rootkey_ecc.pem
ns@ns08:~$ openssl pkeyutl -verify -sigfile root_sign -in root_digest -inkey rootkey_pbkey_ecc.pem -pubin
Signature Verified Successfully
```

Task 2:

Secure Chat App Design

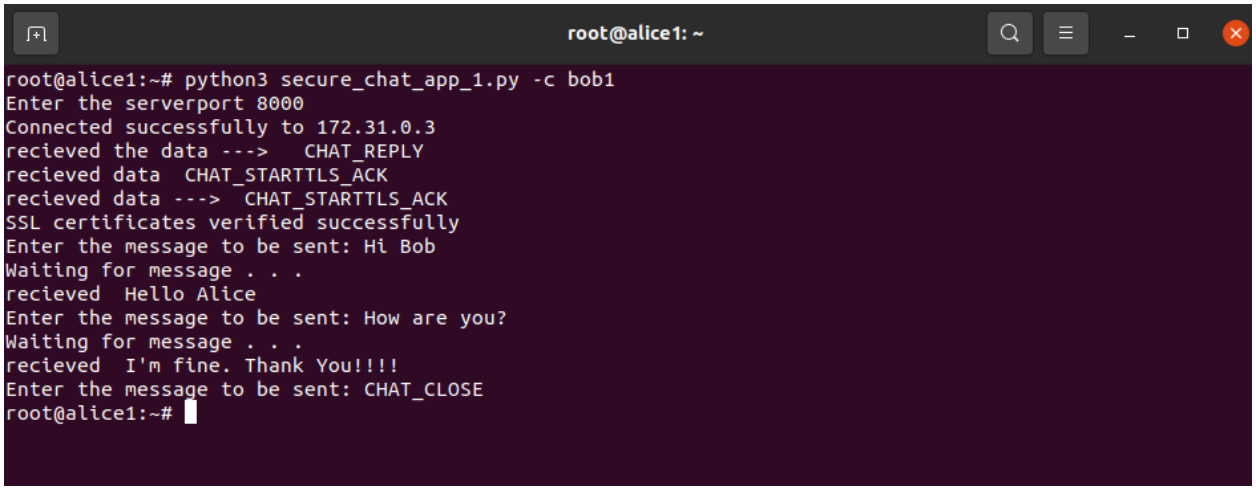
- There are three LXD containers in the VM: allice1, bob1, trudy1. Inside each container all relevant files are present.

- The secure_chat_app.py file is used to establish a secure and reliable connection between server and client which primarily uses TLS 1.3 and TCP protocols. The program uses sockets to communicate among Alice (client) and Bob (server).

- Bob uses secure_chat_app -s to start using the application.
- Alice uses secure_chat_app -c' to start using the application.

Client Side:

- Alice acts as client. The program consists of a class called alice_Client. When secure_chat_app -c is executed this class is called.
- Initially a socket is created for the client. The gethostbyname() function is used to fetch the IP address of the host.
- The start_connection() function then uses connect() to connect to the server. After connecting, Alice sends the chat_hello message to Bob(server).
- If Alice receives back CHAT_STARTTLAS_ACK, it loads the key and certificate into SSL context.
- The wrap_socket() function then wraps the client socket with SSL context and executes the TLS 1.3 handshake with the server.
- If Alice doesn't receive CHAT_STARTTLS_ACK, it continues the connection and keeps on sending messages but in an insecure manner that is without performing TLS 1.3 Handshake.
- When Alice receives chat_close, it closes the TCP and TLS connection and the program terminates.

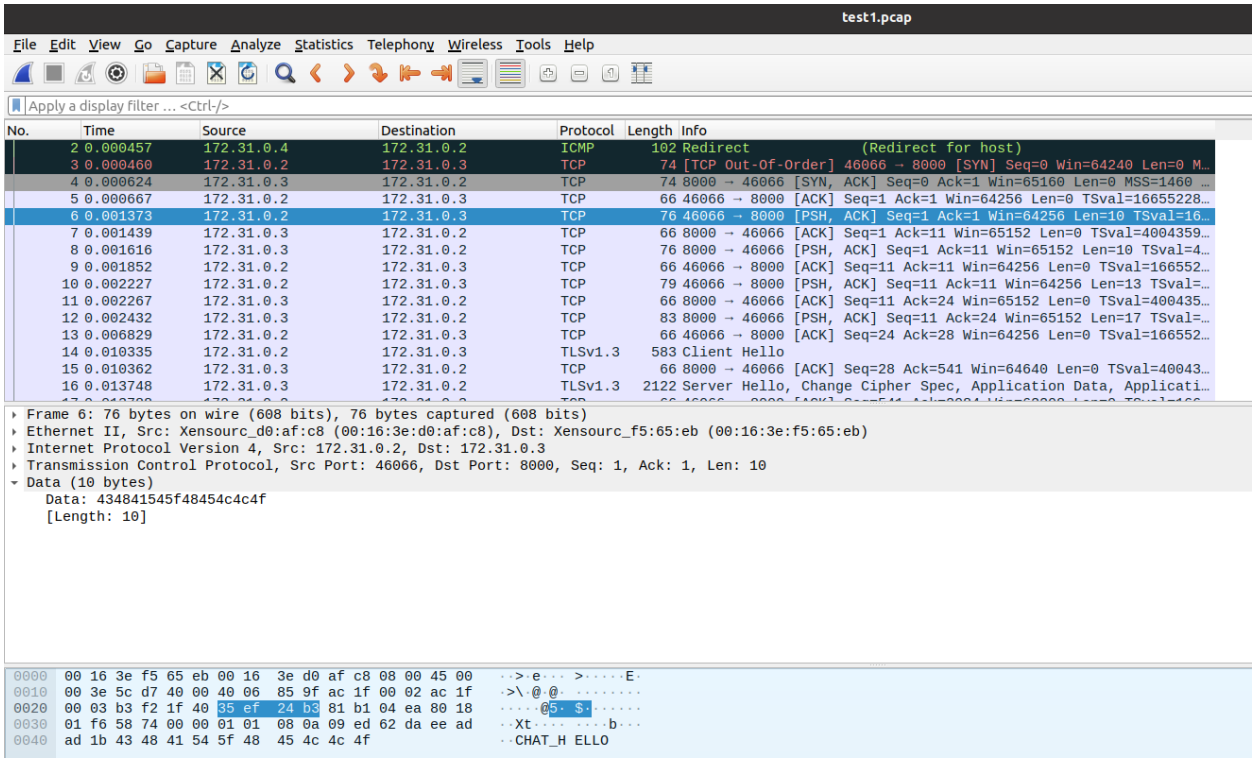


Server Side:

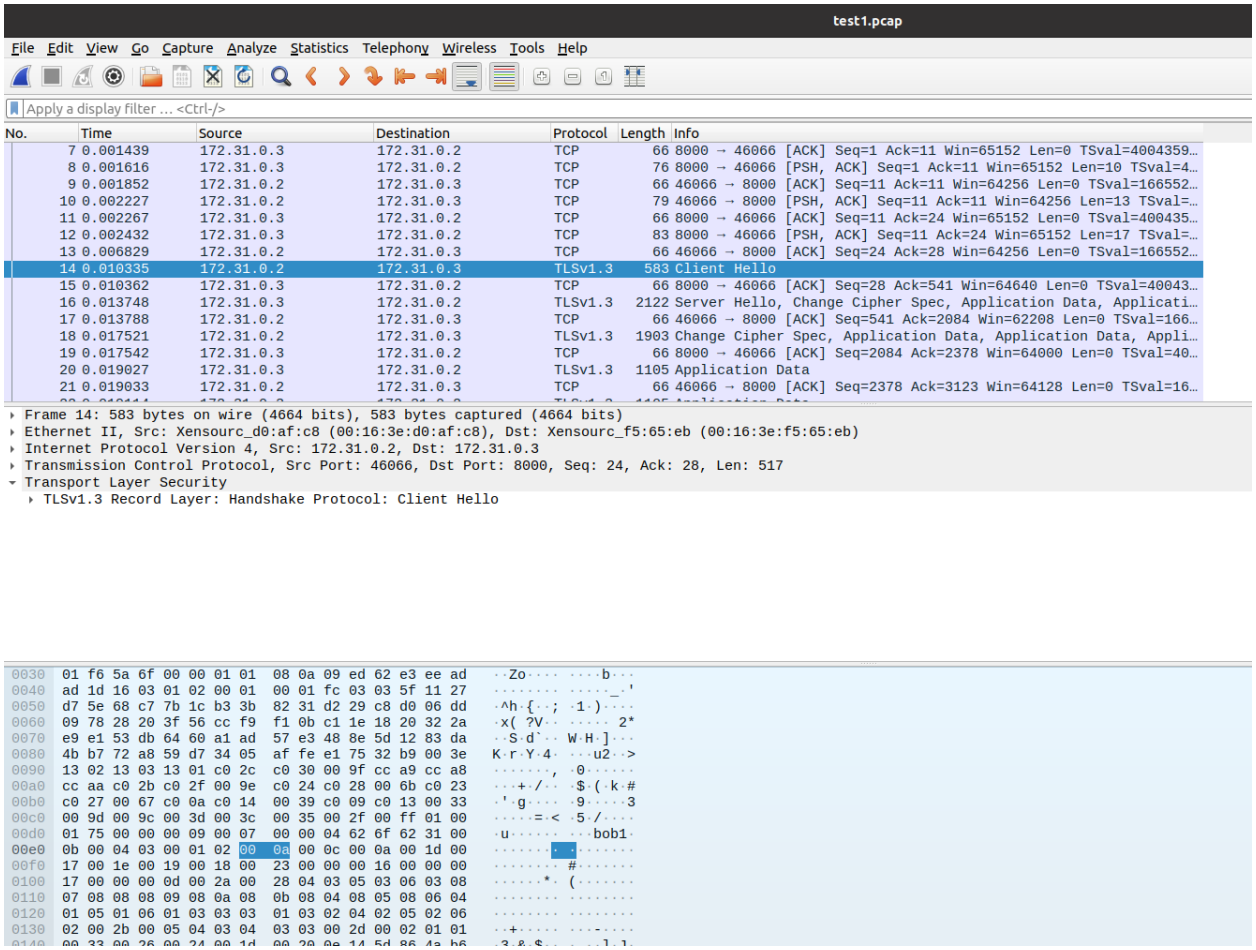
- Bob acts as server. The program consists of a class called bob_Server. When secure_chat_app -s is executed this class is called.
- In the start_connection() function inside the class, a socket is created for the server which is binded with server port and is set to listen mode to wait for client messages.
- After getting connected with the client's address, Bob waits for Alice to send a CHAT_HELLO message, once received it sends back a CHAT_REPLY message to Alice to ensure a secure TCP connection between them.
- Then Bob waits for Alice to send a CHAT_STARTTLS message, once received it sends back CHAT_STARTTLS_ACK message to Alice to perform TLS 1.3 handshake.
- Then Bob loads its keys and certificates into the SSL context, verifies the certificates received from Alice and carries out TLS 1.3 Handshake with Alice in order to establish a secure TLS connection.
- When Bob receives chat_close, it closes the TCP and TLS connection and the program terminates.

```
root@bob1: ~
root@bob1:~# python3 secure_chat_app_1.py -s
Enter the serverport 8000
waiting for client for connection
recieved the connection from client ('172.31.0.2', 46158)
Waiting for the message to be received .....
Received! CHAT_HELLO
Waiting for the message to be received .....
Received! CHAT_STARTTLS
Secure TLS 1.3 pipe Established
Waiting for the message to be received .....
Received! Hi Bob
Enter message to send: Hello Alice
Waiting for the message to be received .....
Received! How are you?
Enter message to send: I'm fine. Thank You!!!!
Waiting for the message to be received .....
Received! CHAT_CLOSE
root@bob1:~#
```

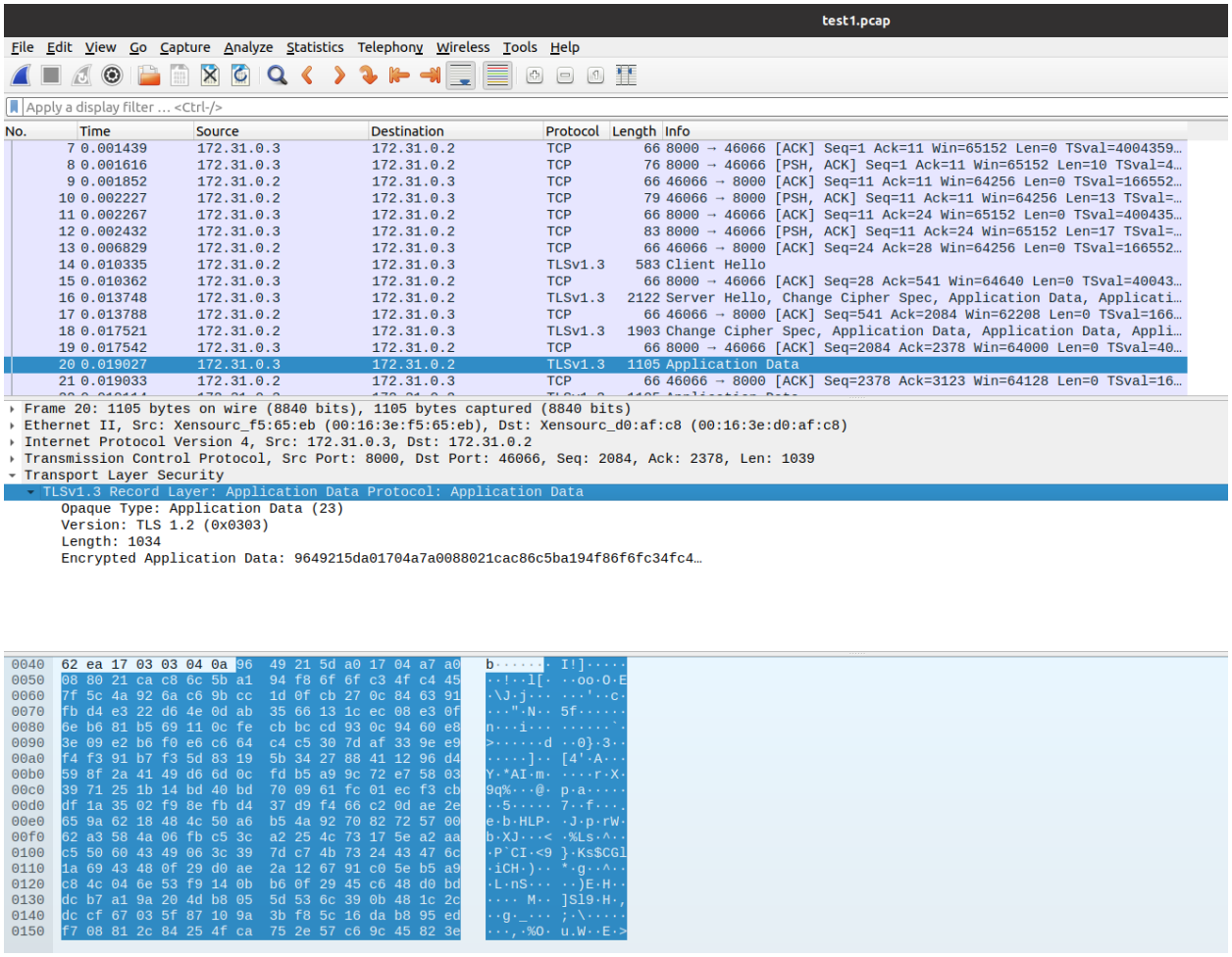
Unencrypted Chat_Hello



TLSv1.3 Handshake between Alice and Bob



Encrypted message after Handshake



Task 3:

Downgrade attack by Trudy

In this task evil Trudy launches a downgrade attack on the communication channel between Alice and Bob.To start this attack following steps need to be followed.

- The /etc/hosts file of both Alice and Bob containers should be poisoned using the command **`bash ~/poison-dns-alice1-bob1.sh.`**
- Run the command `python3 secure_chat_app.py -s` on the Bob container.
- Run the command `python3 secure_chat_interceptor.py -d alice1 bob1` on the Trudy container.
- Run the command `python3 secure_chat_app.py -c` on the Alice container.

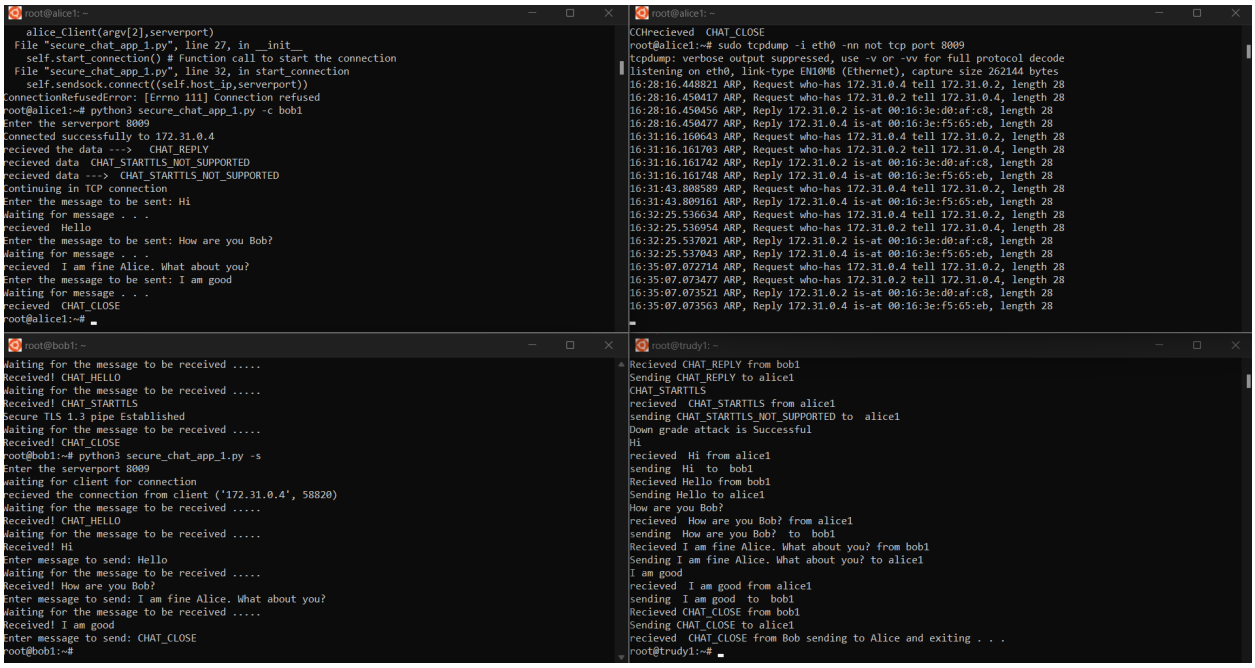
(Note: Enter the same server port number for all the above 3 programs)

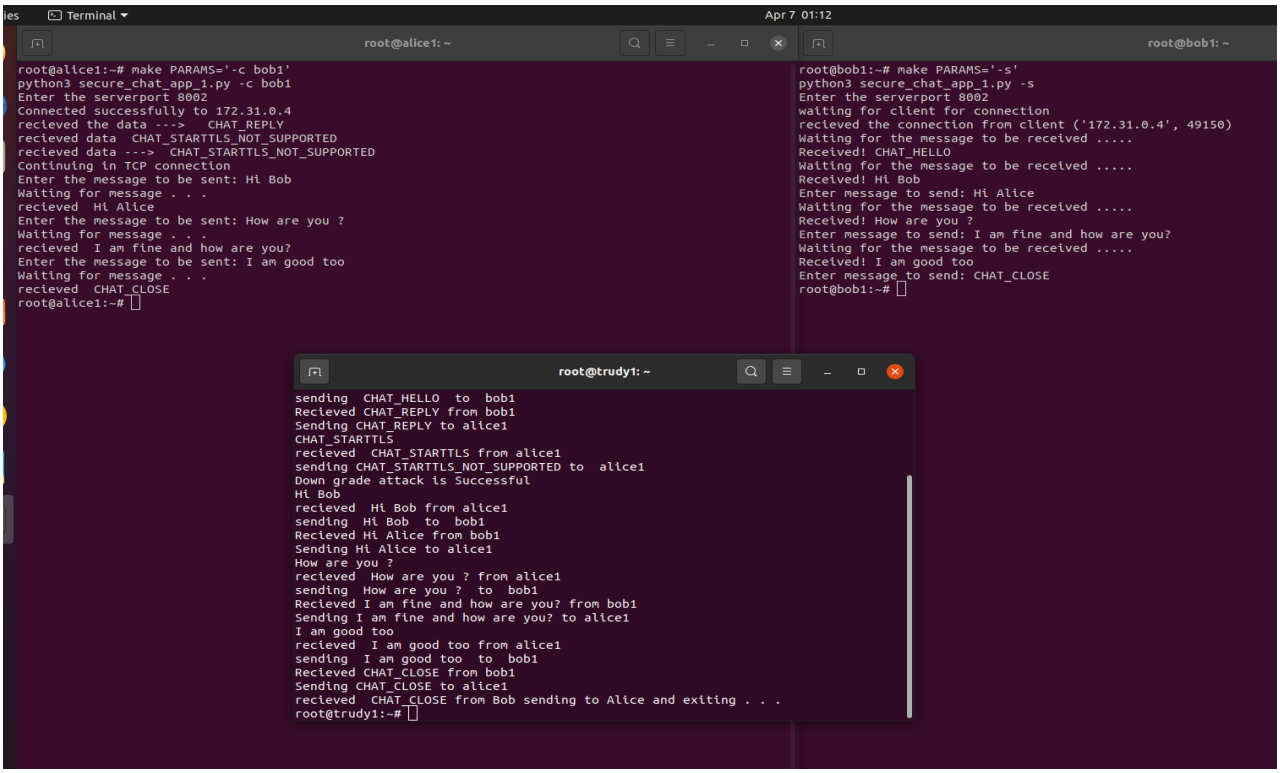
Workflow of downgrade_attack.py:

In this case Trudy intervenes in the connection between Alice and Bob.

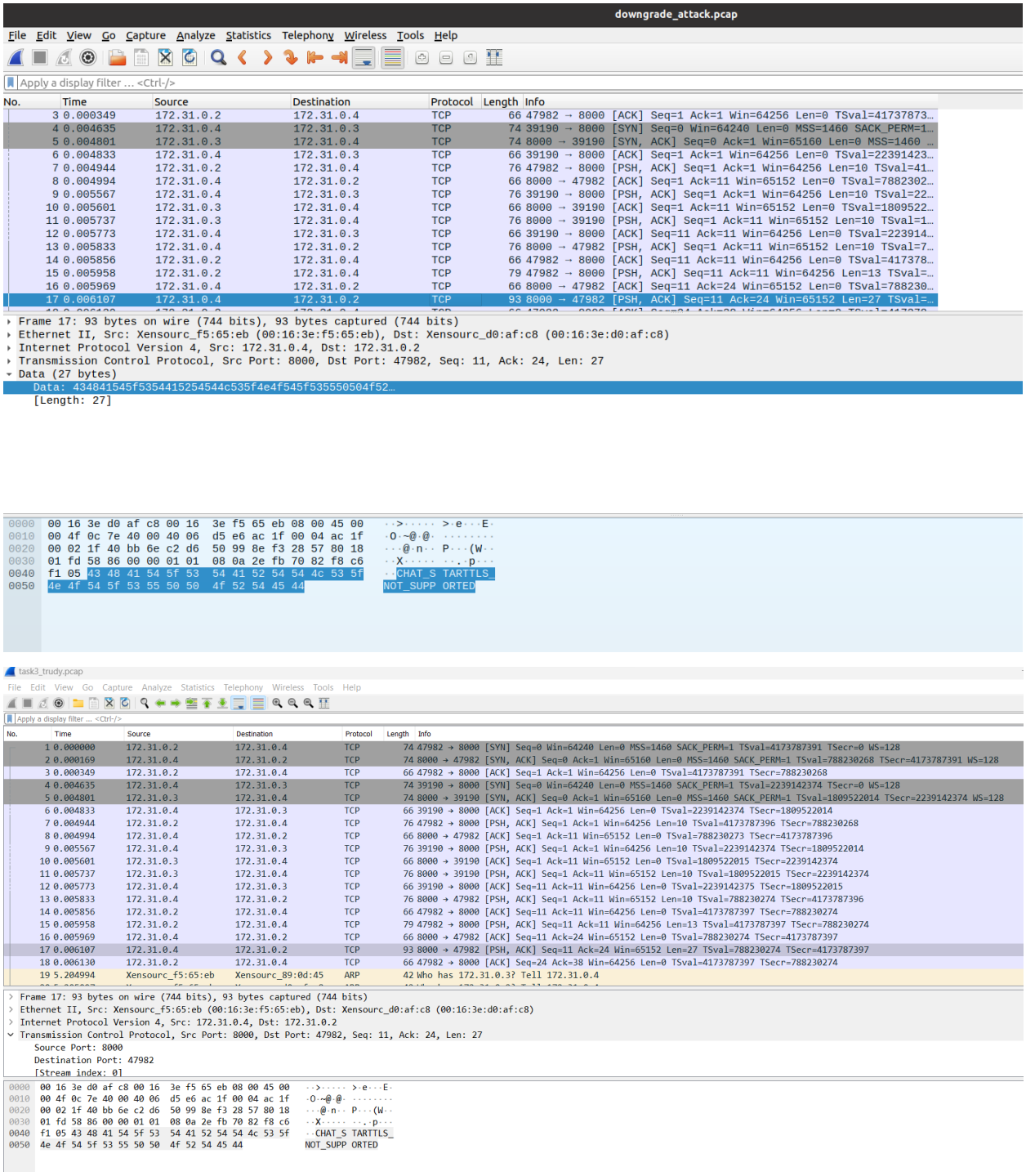
1. So Trudy creates a socket and binds it with the port and listens for any client. Now as the /etc/host file is poisoned Alice will get connected to Trudy. Also a connection is established between Bob and Trudy.
2. So when Trudy receives CHAT_STARTTLS, it sends CHAT_STARTTLS_NOT_SUPPORTED to Alice so that a secure connection is not established.
3. Now the messages can be intercepted by Trudy and can be used for malicious purposes.Downgrade attack is successfully launched.

All the messages sent between Alice and Bob will pass through Trudy.



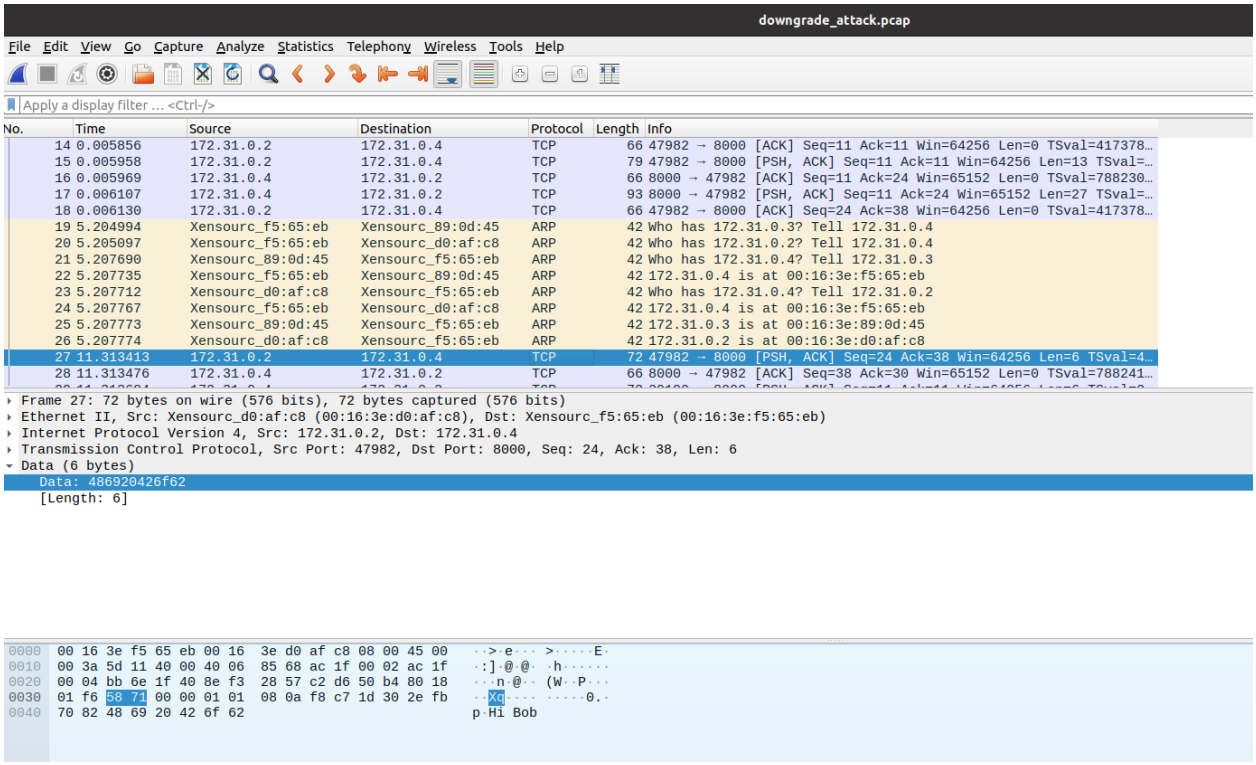


Packet capture of downgrade attack



As we can see that the messages are not encrypted due to the downgrade attack.

<u>Message continued in TCP without Encryption.</u>



Task 4:

In this task first fake CSR of both Alice and Bob are created. The CSRs are signed by the Root private key to get the fake certificate of both Alice and Bob. Same commands are used as in Task 1.

Verification of fake_alice.csr

```
root@trudy1:~# openssl req -text -noout -verify -in fake_alice.csr
verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = IITH, CN = alice1, emailAddress = alice@email.com       Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b6:52:64:26:7d:2a:e3:6e:60:54:1c:3f:96:fd:
                    de:08:40:a1:d7:dc:b0:b4:38:e8:07:b3:81:49:
                    81:c6:67:b9:91:72:2f:38:79:bd:e7:e1:ab:c5:28:
                    0b:e8:34:53:7e:2f:df:fc:0f:ca:bf:b2:57:51:0e:
                    c4:9b:7f:35:8b:1f:d5:d8:4f:b8:a4:12:34:50:0c:
                    07:90:c5:be:17:37:53:b3:24:b0:b9:06:8c:db:76:
                    6e:e3:30:b7:ca:ad:f2:c9:2d:c7:20:93:07:1d:59:
                    69:bd:70:9d:73:69:32:ff:fe:dd:b0:fe:5c:6c:e8:
                    e2:cb:6d:5a:f6:79:1b:5c:97:6a:d6:a5:55:9c:7b:
                    74:e3:bb:f6:75:08:b8:1c:f9:23:55:88:8d:a5:03:
                    d7:17:75:93:a2:92:ba:83:ca:e6:55:47:db:61:28:
                    05:ae:ad:c7:7b:e4:58:b3:e7:6f:d5:2a:dc:3c:cd:
root@trudy1:~#
                    4d:6b:3d:5a:f8:54:90:cc:6c:10:0f:c9:fa:39:e4:
                    52:d3:2b:13:c4:fd:5f:a7:15:23:b8:d5:9a:d5:8b:
                    d2:53:2f:e2:b6:69:80:48:ef:8b:3c:d2:c4:58:1f:
                    fb:43:fe:0e:19:51:4e:6c:1a:ec:0b:27:7d:4a:9d:
                    8d:e9
                Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :alicefakecsr
            challengePassword        :alicefakecsr
    Signature Algorithm: sha256WithRSAEncryption
         14:2f:1b:06:21:4b:28:27:bd:06:b0:29:3f:3e:28:6d:4e:60:
         cb:e0:2c:96:ae:43:95:04:d2:ee:20:21:7d:e0:7a:b0:98:40:
         b0:99:14:74:51:8f:1a:e5:94:ec:61:b1:aa:8b:a8:15:b4:d9:
         c2:27:e4:e2:ae:9e:75:dd:ad:13:2c:b2:45:db:69:53:3d:fe:
         3d:7d:5d:9d:f0:cf:ed:a0:d6:76:4b:9e:65:65:30:33:30:3f:
         6d:9c:7f:bc:70:4d:77:de:d7:62:36:56:fd:d5:61:15:46:8a:
         52:69:be:27:b0:72:10:83:ac:fd:24:1a:eb:a8:66:dc:42:a0:
         0e:b0:30:73:6e:27:1e:34:a4:28:d0:ab:36:c2:b5:b4:f7:e7:
         2c:1e:c3:6f:dd:c9:13:b0:ba:26:47:0b:d6:0d:eb:77:d1:7d:
         cd:6f:7e:e9:04:63:0e:fa:15:48:80:e1:86:ec:bc:47:86:db:
         4c:a8:6d:0e:2a:38:d7:5b:19:ab:31:fd:75:40:a6:bd:19:21:
         44:fd:fd:65:b7:44:03:ec:65:f6:65:99:6d:19:c8:26:2e:d2:
         cf:a1:24:17:3f:28:43:1d:fd:8a:ac:76:c7:d4:d7:96:cc:f0:
```

Verification of fake_bob.csr

```
root@trudy1:~# openssl req -text -noout -verify -in fake_bob.csr
verify OK           4d:6b:3d:5a:f8:54:90:cc:6c:10:0f:c9:fa:39:e4:
Certificate Request:52:d3:2b:13:c4:fd:5f:a7:15:23:b8:d5:9a:d5:8b:
    Data:           d2:53:2f:e2:b6:69:80:48:ef:8b:3c:d2:c4:58:1f:
        Version: 1 (0x0)3:fe:0e:19:51:4e:6c:1a:ec:0b:27:7d:4a:9d:
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = CSE, CN = bob1, emailAddress = bob@gmail.com
        Subject Public Key Info:(0x10001)
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)efakecsr
                Modulus:sword        :alicefakecsr
                    00:b5:71:4a:71:31:fc:65:f1:bd:44:1f:8a:1d:c5:
                    52:b1:65:57:6a:8d:39:b0:54:9a:f2:fb:1e:f3:e9:
                    5d:dd:cb:1c:0b:d1:80:c0:c2:6b:b2:b3:20:d9:d0:
                    94:55:3d:40:7c:6e:75:06:83:d5:0f:b0:df:08:39:
                    9b:6b:ad:68:3f:3d:65:92:2b:85:fe:24:2c:b9:1d:
                    1c:cb:c7:b3:ed:60:b6:98:82:21:fd:58:13:07:b1:
                    ff:09:a9:ef:3a:60:16:37:44:9e:9b:cb:96:1a:2b:
                    06:3b:15:7e:14:91:4b:a7:98:41:a0:d2:f0:b6:22:
                    31:52:84:12:1e:77:d2:0d:21:09:a1:5b:79:b1:97:
                    42:f2:1c:1d:eb:f7:82:a3:fe:ad:11:b1:07:af:75:
                    51:e2:66:d0:08:f3:df:47:7f:9a:e4:e6:7d:65:3e:
                    90:8f:20:20:82:51:23:74:5d:11:31:7f:d8:f3:1e:
                    41:24:f5:75:38:0f:5e:ad:7e:7c:db:e3:c2:73:6b:
                    83:a6:58:35:57:81:2a:e0:ac:da:67:fb:b5:d6:d6:
                    77:f6:ef:2a:b7:02:5f:9b:9b:0c:25:b2:5c:c5:75:
                    b5:ff:f5:f0:5e:45:60:db:c0:d1:d4:b3:04:82:af:
                    c5:00:f5:38:05:29:7a:d8:a7:fe:13:71:de:f4:59:
                    bf:83
                Exponent: 65537 (0x10001)
        Attributes:
            unstructuredName         :IITH
            challengePassword        :bob1
    Signature Algorithm: sha256WithRSAEncryption
         b0:71:88:5c:77:17:9b:00:55:e8:61:a1:1c:9b:25:ff:ed:ab:
         e8:b9:58:d4:67:b4:87:9b:18:af:e9:6c:8e:4f:9d:1f:01:0d:
         43:29:dd:21:88:aa:fd:38:10:7d:da:1b:00:de:9f:a6:05:62:
         74:55:81:29:f4:56:ff:bb:49:7e:d1:a9:52:09:cf:42:59:e8:
         9f:7a:38:48:72:ee:4a:0a:85:ac:50:75:0d:fa:41:72:bf:11:
         c8:79:c6:33:84:d1:9e:82:18:36:71:6c:a3:f6:aa:d0:37:eb:
         d3:a7:48:e8:a4:8f:8f:3c:39:39:95:cd:23:33:bc:0f:6f:b5:
         37:5d:5c:ea:6f:e6:16:6b:83:8c:d1:ab:f7:e2:73:6a:7e:e1:
         0e:8b:fb:ea:99:a8:cc:8e:e0:78:ff:20:91:22:74:7c:7e:5d:
         11:a4:f9:30:24:9d:47:82:46:d9:9b:b8:bf:17:28:60:13:b1:
         ad:27:e6:76:38:74:52:42:60:95:45:ab:18:c9:dc:f7:24:90:
         9e:d9:82:b7:8b:49:c2:35:a0:79:f7:d4:98:ce:11:c0:35:43:
         b9:7f:d9:02:25:5e:6c:cb:e2:d3:7a:14:7c:7c:81:fb:ef:81:
         62:97:b1:97:e6:fb:59:55:5b:ce:7f:1b:55:37:b0:57:e3:7f:
         b9:c0:82:4d
```

Verification of fake_alice.crt:

```
root@trudy1:~# openssl verify -verbose -CAfile root.crt fake_alice.crt
fake_alice.crt: OK
root@trudy1:~#
```

Verification of fake_bob.crt

```
root@trudy1:~# openssl verify -verbose -CAfile root.crt fake_bob.crt
fake_bob.crt: OK
root@trudy1:~#
```

Active MITM attack by Trudy

To launch this attack, we first run the script `bash ~/poison-dns-alice1-bob1.sh` to poison the etc/hosts files of Alice and Bob containers.

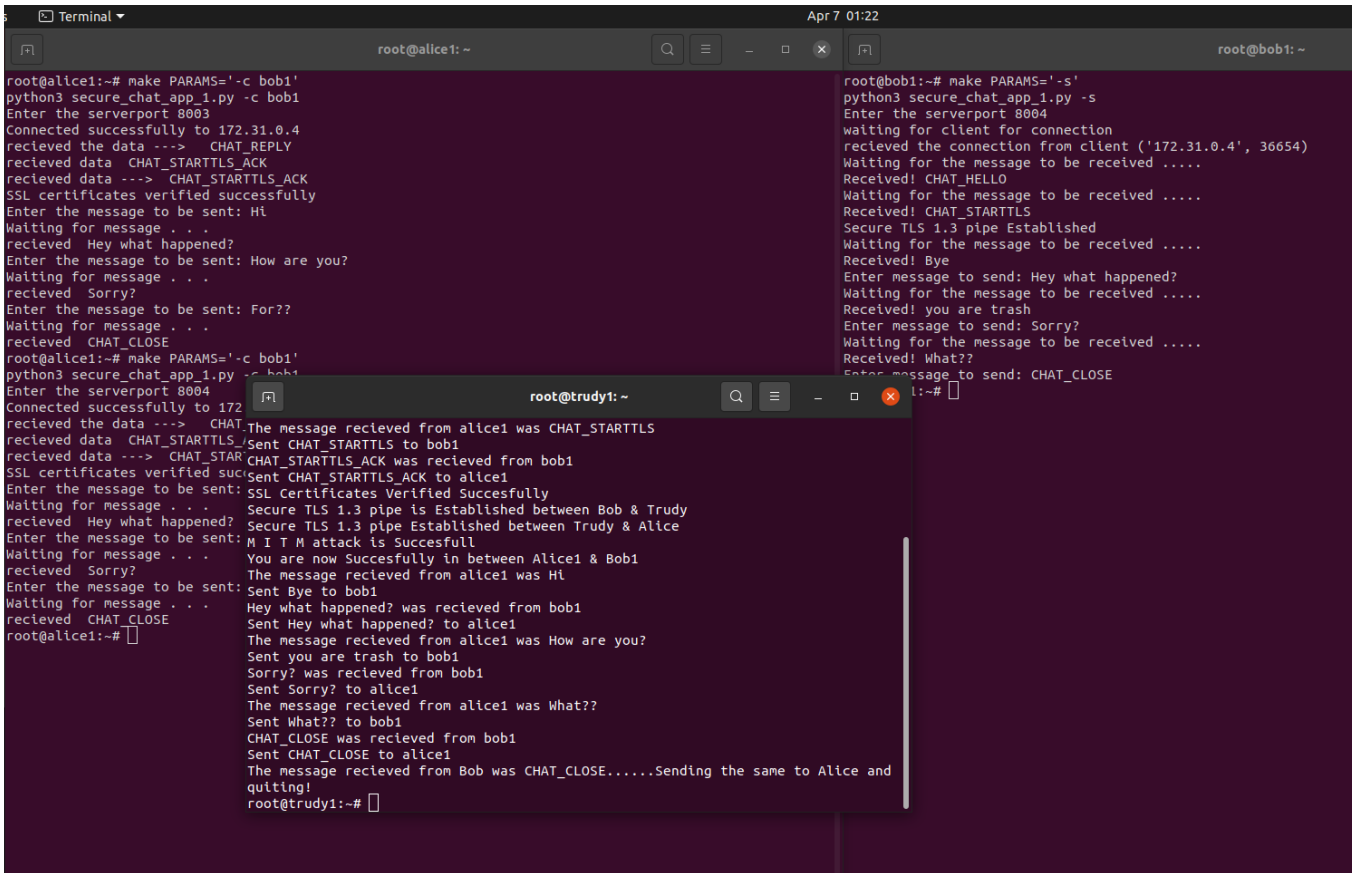
- From inside the Bob1 container, run the python script secure_chat_app_1.py as `make PARAMS='-s'`
- From inside the Trudy container we run the python script main_securechat_interceptor.py with an argument -m to launch the MITM attack. The command is `make PARAMS='-m alice1 bob1'`
- From inside the Alice1 container, run the python script secure_chat_app_1.py as `make PARAMS='-c bob1'`

(Note: Enter the same server port number for all the above 3 programs)

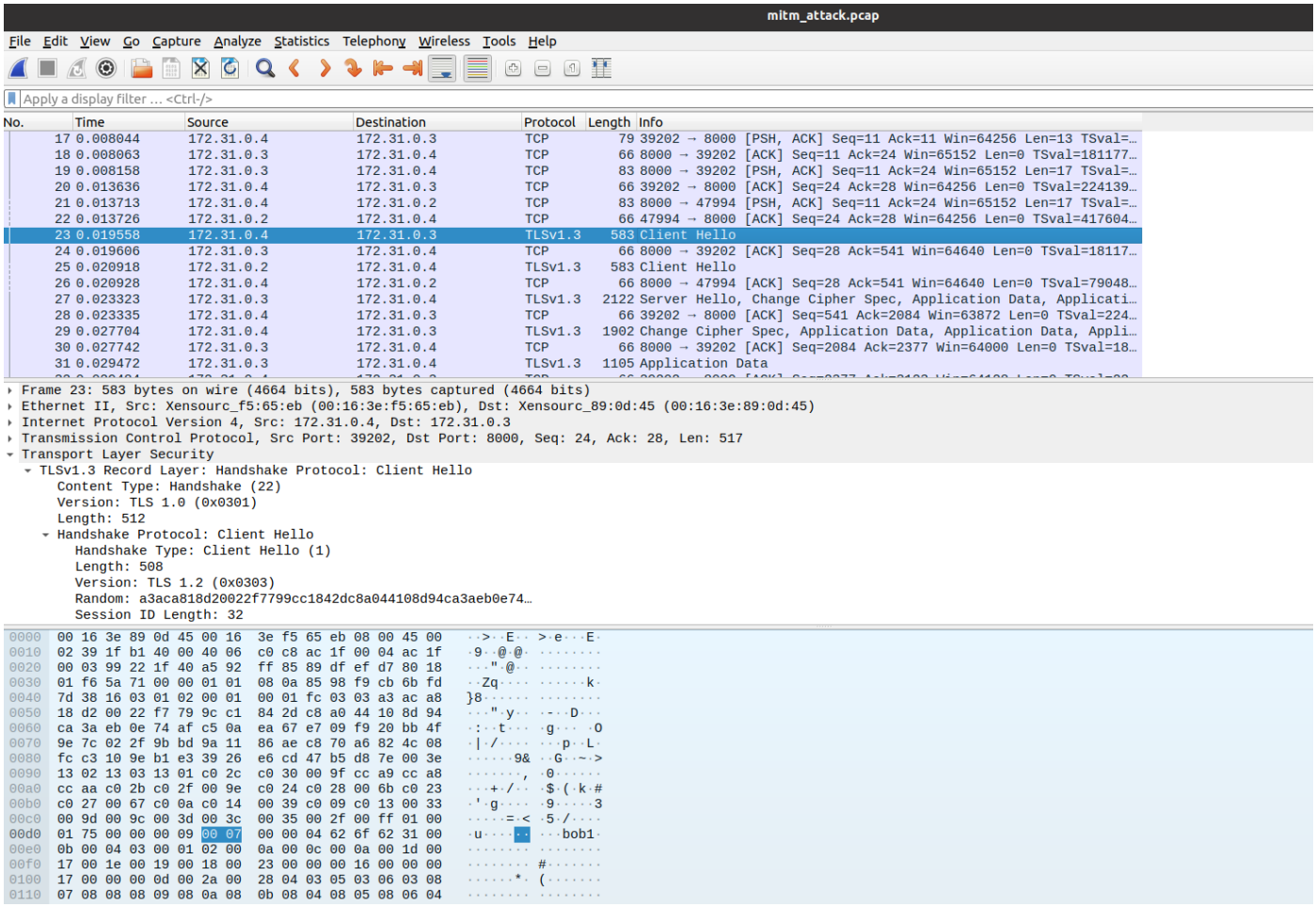
Workflow of the MITM_attack.py file

1. The arguments from the command line give us the clientname and the servername. With the help of these we create a pseudo connection i.e we connect Alice and Bob with a pseudo sender and receiver sockets.
2. The exchange of chat headers starts. `CHAT_HELLO` is being sent by Alice. Upon receiving this, Trudy forwards the same message to Bob. Then Trudy after receiving `CHAT_REPLY` from Bob, she simply forwards this to Alice.

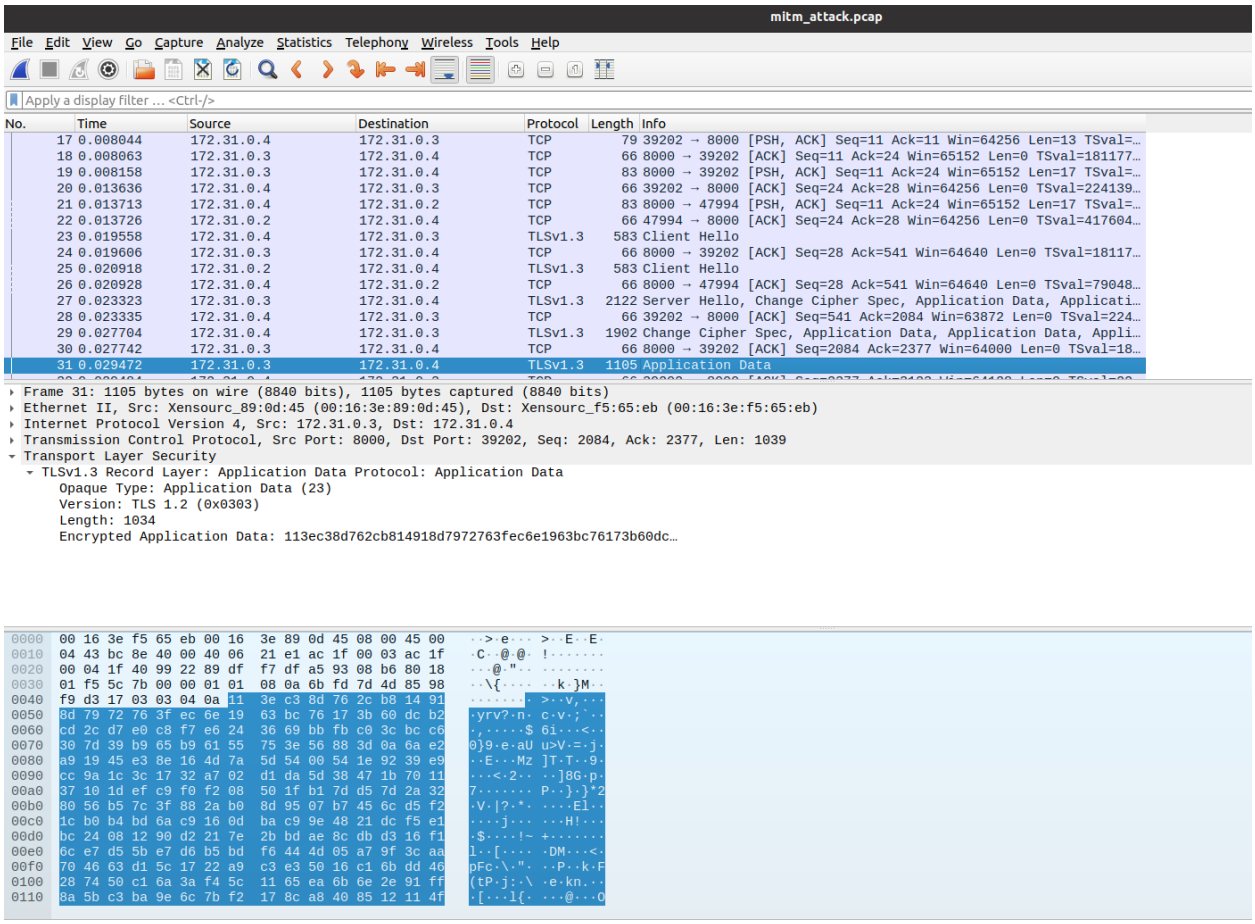
3. Now when Trudy receives `CHAT_STARTTLS` message from Alice and forwards it to Bob.Once she receives back `CHAT_STARTTLS_ACK` from Bob , Trudy sets up two TLS connections:
    a. Trudy and Bob TLS connection - Fake certificates and keys of Bob are verified and the handshake is done!TLS pipe is established .
    b. Trudy and Alice TLS connection - Fake certificates and keys of Alice are verified and the handshake is done!TLS pipe is established .
4. Once the handshake is successfully done between both the parties, Alice and Bob exchange messages between themselves which are overheard by Trudy and tampered as well.
5. In order to demonstrate the tampering of messages that are sent by Alice , we have used 3 instances :
    a. If Alice sends `"How are you?"` to Bob, Trudy receives the message and changes it to “you are trash” and sends it to Bob.
    b. Similarly, you can try sending other messages such as `"God bless you”` and `"Hi"` and see for yourself the tampering that Trudy does before sending it to Bob.
6. As mentioned in the assignment, `CHAT_CLOSE` is the message that denotes the closure of the chat session and the TLS connection. So upon receiving that message we close the TLS and TCP sockets and end the session.


Terminal
Apr 7 01:22
root@alice1: ~
root@alice1:~# make PARAMS='-c bob1'
python3 secure_chat_app_1.py -c bob1
Enter the serverport 8003
Connected successfully to 172.31.0.4
recieved the data ---> CHAT_REPLY
recieved data CHAT_STARTTLS_ACK
recieved data ---> CHAT_STARTTLS_ACK
SSL certificates verified successfully
Enter the message to be sent: Hi
Waiting for message . . .
recieved Hey what happened?
Enter the message to be sent: How are you?
Waiting for message . . .
recieved Sorry?
Enter the message to be sent: For??
Waiting for message . . .
recieved CHAT_CLOSE
root@alice1:~# make PARAMS='-c bob1'
python3 secure_chat_app_1.py -c bob1
Enter the serverport 8004
Connected successfully to 172
recieved the data ---> CHAT
recieved data CHAT_STARTTLS_
recieved data ---> CHAT_STAR
SSL certificates verified suc
Enter the message to be sent:
Waiting for message . . .
recieved Hey what happened?
Enter the message to be sent:
Waiting for message . . .
recieved Sorry?
Enter the message to be sent:
Waiting for message . . .
recieved CHAT_CLOSE
root@alice1:~#
root@bob1: ~
root@bob1:~# make PARAMS='-s'
python3 secure_chat_app_1.py -s
Enter the serverport 8004
waiting for client for connection
recieved the connection from client ('172.31.0.4', 36654)
Waiting for the message to be received .....
Received! CHAT_HELLO
Waiting for the message to be received .....
Received! CHAT_STARTTLS
Secure TLS 1.3 pipe Established
Waiting for the message to be received .....
Received! Bye
Enter message to send: Hey what happened?
Waiting for the message to be received .....
Received! you are trash
Enter message to send: Sorry?
Waiting for the message to be received .....
Received! What??
Enter message to send: CHAT_CLOSE
1:~#
root@trudy1: ~
The message recieved from alice1 was CHAT_STARTTLS
Sent CHAT_STARTTLS to bob1
CHAT_STARTTLS_ACK was recieved from bob1
Sent CHAT_STARTTLS_ACK to alice1
SSL Certificates Verified Succesfully
Secure TLS 1.3 pipe is Established between Bob & Trudy
Secure TLS 1.3 pipe Established between Trudy & Alice
M I T M attack is Succesfull
You are now Succesfully in between Alice1 & Bob1
The message recieved from alice1 was Hi
Sent Bye to bob1
Hey what happened? was recieved from bob1
Sent Hey what happened? to alice1
The message recieved from alice1 was How are you?
Sent you are trash to bob1
Sorry? was recieved from bob1
Sent Sorry? to alice1
The message recieved from alice1 was What??
Sent What?? to bob1
CHAT_CLOSE was recieved from bob1
Sent CHAT_CLOSE to alice1
The message recieved from Bob was CHAT_CLOSE.....Sending the same to Alice and quiting!
root@trudy1:~#

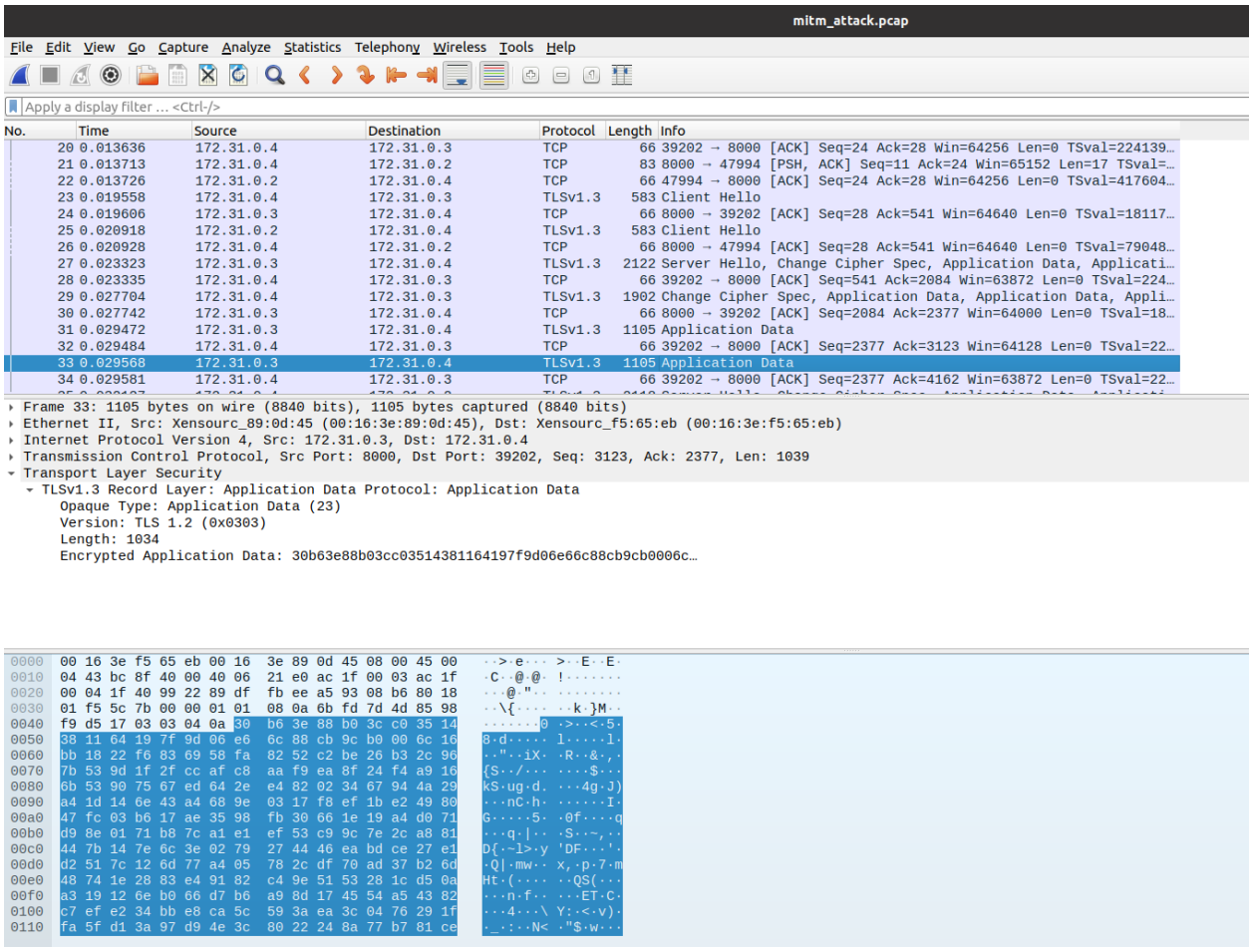
Two TLSv 1.3 Pipes created (Two Handshake Done)



Message encrypted between Trudy and Bob in TLS



Message encrypted between Trudy and Alice in TLS



Bonus:

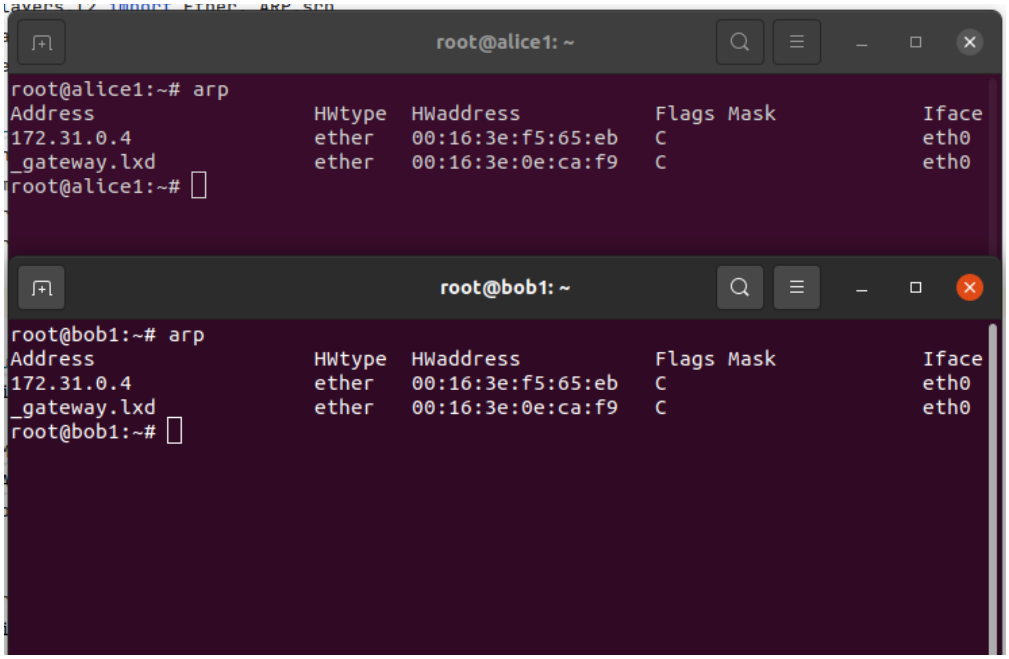
In this task , we have implemented ARP spoofing which is a man in the middle attack . Trudy in this task tries to fool both Alice and Bob by adding her MAC in the arp tables of each other and hence successfully making them connect to her network.

Workflow of ARP_poisioning.py

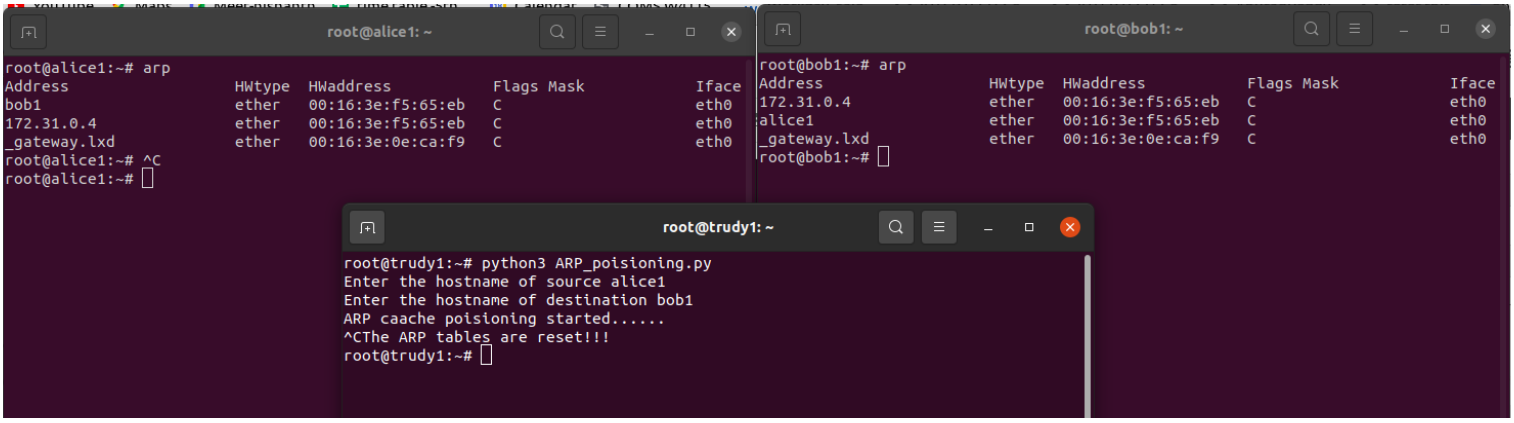
1. Once the hostnames of two hosts are entered, we call the function spoof which spoofs the MAC of Alice to Bob and that of Bob to Alice.

2. The ARP tables of both the hosts will be manipulated i.e the arp of alice will have trudy spoofing bob and the arp of bob will have trudy spoofing alice.
3. Once the execution is interrupted, the tables are reset to their original state.

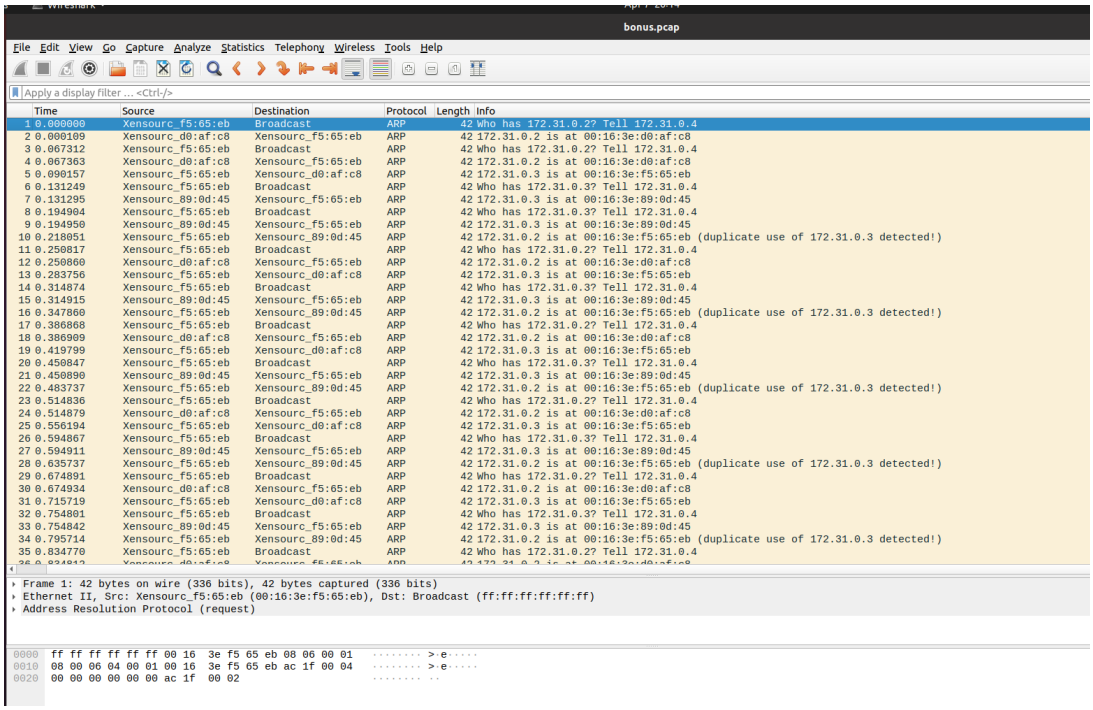
Original state



ARP poisoning



PCAP file screenshot



PLAGIARISM STATEMENT <Include it in your report>

We certify that this assignment/report is our own work, based on our personal study and/or research and that we have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other kind of document, electronic or personal communication. We also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that we have not copied in part or whole or otherwise plagiarized the work of other students and/or persons. We pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, We understand my responsibility to report honor violations by other students if we become aware of it.

Names:Nisha, Koustav, Priyansha
Date: April 7,2022
Signature: M,Choudhury,Tiwari

References:

1. OpenSSL Cookbook: Chapter 1. OpenSSL Command Line (feistyduck.com)
2. /docs/man1.1.1/man3/index.html (openssl.org)
3. OpenSSL client and server from scratch, part 1 – Arthur O'Dwyer – Stuff mostly about C++ (quuxplusone.github.io)
4. ssl — TLS/SSL wrapper for socket objects — Python 3.9.2 documentation
5. Secure programming with the OpenSSL API – IBM Developer
6. Simple TLS Server - OpenSSLWiki
7. The /etc/hosts file (tldp.org)
8. PowerPoint Presentation (owasp.org)
9. SEED Project (seedsecuritylabs.org)