

## Network Security Assignment No.7: Secure chat using openssl and MITM attacks

### Group members:

Name: Hrishikesh Hemke

Roll No.: CS23MTECH14003

Name: Dindorkar Mayuresh Rajesh

Roll No.: CS23MTECH14007

Name: Shrenik Ganguli

Roll No.: CS23MTECH14014

### Note:

1. The entire code is implemented in **C programming language** because it offers low-level access to networking APIs, socket operations, and network protocols in great detail.
2. C is more environment-friendly than C++.

### TASK 1: Generate Keys and Certificates

#### Step 1: Generating the keys (.pem)

##### Root CA:

For generating private key:

```
openssl ecparam -genkey -name brainpoolP512r1 -out root_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl ecparam -genkey -name brainpoolP512r1 -out root_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl ec -in root_private_key.pem -text -noout
read EC key
Private-Key: (512 bit)
priv:
9c:b7:90:96:2e:08:c1:b1:ef:3d:d4:e2:c2:69:
25:99:e6:e4:fb:ef:a8:c7:47:fc:e7:bb:dc:c9:6b:
78:c8:48:cf:d2:fd:d6:b3:73:4c:65:b9:ff:f9:e8:
05:c8:1c:43:e2:06:a1:5f:21:3f:3c:96:26:db:7f:
af:8a:8a:6d
pub:
04:2f:04:ea:e9:61:ae:29:35:9f:8f:14:64:5b:9a:
3c:49:19:f2:e5:8b:a9:fe:60:bf:46:af:66:72:49:
89:46:44:49:af:49:0f:b1:1e:25:33:1c:87:9b:0e:
c5:c0:46:02:23:8f:3a:51:ab:96:1c:8d:d1:ef:9f:
3c:a1:6b:08:79:a1:db:17:11:57:24:27:fb:50:b6:
30:a2:61:5c:2f:3c:4c:d0:f8:c2:36:71:3f:61:64:
38:9c:4c:a3:57:a3:d8:fb:3a:8c:f3:ae:d8:b5:f1:
24:7f:e9:41:a1:f0:ee:1b:31:fd:63:f0:bf:0a:d5:
dc:17:87:a0:af:9d:81:45:7c
ASN1 OID: brainpoolP512r1
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

For extracting public key:

```
openssl ec -in root_private_key.pem -pubout -out root_public_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl ec -in root_private_key.pem -pubout -out root_public_key.pem
read EC key
writing EC key
-----BEGIN PUBLIC KEY-----
MIIBMBQGByqGSM49AgEGCSsKAwMCCEBDQOBggAELwTq6WGuKTWfjxRkW5o8SRny
5Yup/mC/Rq9mcknJRKJR0kPsR4lMxyHmw7FwEYCI486UsuWhI3R75880wsIeahb
FxFXJCf7ULYwomFcLzxM0PjCNhE/YWQ4nEyjV6PY+zqM867YtfEkf+LBofDuGzH9
Y/C/atXcF4egr52BRXw=
-----END PUBLIC KEY-----
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

## Intermediate CA:

```
openssl genpkey -out int_private_key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl genpkey -out int_private_key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:4096
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl rsa -in int_private_key.pem -text -noout
Private-Key: (4096 bit, 2 primes)
modulus:
00:cd:5a:86:4b:ce:46:f7:74:1e:c8:cb:55:26:43:
66:1b:d7:8c:47:6e:c9:b7:96:08:2b:45:a0:cb:0f:
5e:43:94:2b:c3:58:72:65:c1:d9:28:53:d7:ac:02:
0f:00:98:f3:0b:51:bb:27:bd:26:7b:6a:aa:5c:52:
d5:5f:7a:3a:42:f5:59:ee:e9:c4:64:35:64:a3:bc:
af:27:42:7c:5f:e5:a8:c8:a8:8f:e8:61:30:4f:68:
93:2a:00:26:80:88:a7:86:23:f1:99:d4:c5:06:b9:
14:08:f7:96:d0:ed:f6:a2:41:34:78:3d:18:98:b3:
d2:15:6d:a5:1d:ce:c2:39:5f:f5:61:6c:fc:97:48:
b3:84:3b:30:b7:c7:d2:6c:12:38:7c:6a:c2:f6:da:
3e:82:d0:66:8d:15:05:0d:ee:2e:e0:40:5e:c7:ff:
4e:34:fc:30:c0:3d:95:b6:fd:97:67:0a:d7:1d:50:
```

## Alice:

```
openssl genpkey -out alice_private_key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:1024
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl genpkey -out alice_private_key.pem -algorithm RSA -pkeyopt rsa_keygen_bits:1024
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl rsa -in alice_private_key.pem -text -noout
Private-Key: (1024 bit, 2 primes)
modulus:
00:a9:1d:9d:4c:ed:6a:9f:fa:71:83:21:96:42:00:
c3:cc:42:02:b0:8f:d8:24:d0:1d:db:21:d6:55:1c:
cc:95:2e:9e:9b:a1:06:43:b4:bc:ea:d2:b0:9f:2f:
aa:9c:b2:5e:a7:8e:27:84:b6:12:ef:0f:a0:7b:fc:
6d:71:5c:40:06:fc:fc:3b:d9:1a:81:4f:60:ca:0d:
7c:dc:fc:30:94:6a:2f:34:el:66:bb:59:bc:b2:4e:
fc:33:a7:b0:38:d9:da:a6:8e:25:f6:a2:9a:4b:98:
f6:36:36:35:55:f9:5a:09:20:f9:a9:3b:59:48:85:
80:67:7d:bb:bf:bb:8f:94:a1
publicExponent: 65537 (0x10001)
privateExponent:
08:c5:5f:6e:28:f6:19:bb:fc:7d:27:95:ca:70:8d:
b7:1f:49:52:6c:f3:71:dd:55:5f:13:1f:31:a4:df:
bb:71:02:8a:e1:cc:18:60:dc:57:57:7b:f1:95:a0:
4c:a6:37:a0:8b:0f:e3:a3:29:d2:3a:46:41:87:f3:
09:29:16:ac:f0:84:aa:dc:7f:ff:2a:4e:1f:fd:b3:
fe:45:1a:87:e5:0c:bb:3c:db:cb:f2:99:4a:65:90:
3f:57:c6:a1:24:b8:a5:f3:f3:67:ea:f0:fd:
cf:ea:6d:d1:d2:65:f1:87:28:1b:4a:53:9d:5a:92:
30:6c:e4:2d:2d:df:f5:dd
```

## Bob:

```
openssl ecpparam -name prime256v1 -genkey -noout -out bob_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl ecpparam -name prime256v1 -genkey -noout -out bob_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl ec -in bob_private_key.pem -text -noout
read EC key
Private-Key: (256 bit)
priv:
b6:ba:0c:35:e5:b0:a0:00:93:2d:74:3e:62:09:
7f:f6:f5:2b:e5:bc:c3:5c:c3:23:38:4d:40:77:f3:
e5:36
pub:
04:56:32:6a:a1:4d:e9:bc:d8:90:bc:f7:08:b3:f6:
b8:4c:cb:3b:f8:33:b5:03:31:72:6e:e8:2c:36:c6:
cf:b0:e9:3e:6c:86:5e:e4:c5:83:46:f8:b6:e7:e8:
3b:c5:f4:c8:0a:88:84:c7:29:ae:fc:49:d7:2a:aa:
9f:63:d6:13:55
ASN1 OID: prime256v1
NIST CURVE: P-256
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

## **Step 2: Creating the configuration(.cnf) and extension files(.txt)**

### **RootCA:**

#### **Config file: root.cnf**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ cat root.cnf
[req]
prompt = no
distinguished_name = root_ca
input_password = root

[root_ca]
CN = iTS ROOT R1
emailAddress = rootCA@gmail.com
O = iTS ROOT R1
L = Hyderabad
C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

#### **Extension file: root.txt**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ cat root.txt
authorityKeyIdentifier = keyid:always,issuer:always
basicConstraints      = CA:TRUE
keyUsage              = critical,keyCertSign,cRLSign
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

### **Intermediate CA:**

#### **Config file: int.cnf**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat int.cnf
[req]
prompt = no
distinguished_name = int_ca
input_password = int

[int_ca]
CN = iTS CA 1R3
emailAddress = intCA@gmail.com
O = iTS CA 1R3
L = Nashik
C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

#### **Extension file: int.txt**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat int.txt
authorityKeyIdentifier = keyid,issuer
basicConstraints = CA:TRUE
keyUsage = critical,keyCertSign,cRLSign
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

**Alice:**

**Config file: alice.cnf**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ cat alice.cnf
[req]
prompt = no
distinguished_name = alice_dn
input_password = alice

[alice_dn]
CN = Alice1.com
emailAddress = Alice1@gmail.com
O = Alice1
L = Nagpur
C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

**Extension file: alice.txt**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ cat alice.txt
authorityKeyIdentifier = keyid,issuer
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

**Bob:**

**Config file: bob.cnf**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ cat bob.cnf
[req]
prompt = no
distinguished_name = bob_dn
input_password = bob

[bob_dn]
CN = Bob1.com
emailAddress = Bob1@gmail.com
O = Bob1
L = Kolkata
C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

**Extension file: bob.txt**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ cat bob.txt
authorityKeyIdentifier = keyid,issuer
basicConstraints      = CA:FALSE
keyUsage              = digitalSignature,nonRepudiation,keyEncipherment,dataEncipherment
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

### **Step 3: Generating the Certificate Signing Request (.csr):**

#### **Root CA:**

```
openssl req -new -config root.cnf -key root_private_key.pem -out root.csr
```

```
mayuresh@mayuresh-HP-Laptop: ~/Desktop/secure_chat_submission/root$ openssl req -new -config root.cnf -key root_private_key.pem -out root.csr
mayuresh@mayuresh-HP-Laptop: ~/Desktop/secure_chat_submission/root$ openssl req -text -noout -in root.csr
Certificate Request:
Data:
Version: 1 (0x0)
Subject: CN = iTS ROOT R1, emailAddress = rootCA@gmail.com, O = iTS ROOT R1, L = Hyderabad, C = IN
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (512 bit)
pub:
        04:2f:04:ea:e9:61:ae:29:35:9f:8f:14:64:5b:9a:
        3c:49:19:f2:e5:8b:a9:fe:60:bf:46:af:66:72:49:
        89:46:44:49:af:49:0f:b1:1e:25:33:1c:87:9b:0e:
        c5:c0:46:02:23:8f:3a:51:ab:96:1c:8d:d1:ef:9f:
        3c:a1:6b:08:79:a1:db:17:11:57:24:27:fb:50:b6:
        30:a2:61:5c:2f:3c:4c:d0:f8:c2:36:71:3f:61:64:
        38:9c:4c:a3:57:a3:d8:fb:3a:8c:f3:aed:8:b5:f1:
        24:7f:e9:41:a1:f0:ee:1b:31:fd:63:f0:bf:6a:d5:
        dc:17:87:a0:af:9d:81:45:7c
        ASN1 OID: brainpoolP512r1
Attributes:
    (none)
Requested Extensions:
Signature Algorithm: ecdsa-with-SHA256
Signature Value:
30:81:84:02:40:1a:85:f0:8b:78:e0:65:65:65:65:ad:65:94:
f2:e4:09:22:7d:48:14:a1:59:ec:8a:cc:54:53:b1:0f:4b:70:
98:e9:ad:97:6a:9e:e8:f1:ad:c7:03:84:dd:92:e1:f4:62:c9:
32:a0:91:cc:74:74:ce:ac:42:c2:6e:98:fe:96:4c:02:40:10:
1a:40:03:01:f7:0e:db:0b:45:2a:a0:69:62:fa:19:10:e4:
55:d0:75:4e:14:81:fd:e0:9f:8b:ce:fb:7c:ec:6e:fd:c9:ec:
36:6e:19:f9:c3:a7:a7:0d:16:4f:a8:20:05:85:23:41:c6:a5:
c6:cd:10:5a:cb:1e:ea:97:b0
mayuresh@mayuresh-HP-Laptop: ~/Desktop/secure_chat_submission/root$
```

#### **Intermediate CA:**

```
openssl req -new -config int.cnf -key int_private_key.pem -out int.csr
```

```
mayuresh@mayuresh-HP-Laptop: ~/Desktop/secure_chat_submission/int$ openssl req -new -config int.cnf -key int_private_key.pem -out int.csr
mayuresh@mayuresh-HP-Laptop: ~/Desktop/secure_chat_submission/int$ openssl req -text -noout -in int.csr
Certificate Request:
Data:
Version: 1 (0x0)
Subject: CN = iTS CA IR3, emailAddress = intCA@gmail.com, O = iTS CA IR3, L = Nashik, C = IN
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (4096 bit)
Modulus:
        00:cd:5a:86:4b:ce:46:f7:74:1e:c8:cb:55:26:43:
        c6:1b:d7:8c:47:6e:c9:b7:96:08:2b:45:a0:cb:0f:
        5e:43:94:2b:c3:58:73:65:1:d8:28:53:d7:ac:02:
        0f:00:98:f3:0b:51:bb:27:bd:26:7b:6a:aa:5c:52:
        d5:5f:7a:3a:42:f5:59:ee:e9:c4:64:35:64:a3:bc:
        af:27:42:7c:5f:e5:a8:c8:a8:8f:e8:61:30:4f:68:
        93:2a:00:26:80:88:a7:86:23:f1:99:d4:c5:06:b9:
        14:08:f7:96:d0:ed:f6:a2:41:34:78:3d:18:98:63:
        d2:15:6d:a5:1d:ce:c2:39:f5:f5:61:6c:fc:97:48:
        bb:84:3b:30:b7:c7:d2:6c:12:38:7c:6a:c2:f0:da:
        36:82:d0:66:8d:15:05:0d:ee:2e:e0:40:5e:c7:ff:
        4e:34:fc:30:c0:3d:95:b6:fd:97:67:0a:d7:1d:50:
        ed:db:18:cd:ce:46:0c:bc:32:73:70:8e:4a:76:f7:
        82:ba:c2:8e:27:e6:01:b6:9c:a6:6f:24:0e:a6:a8:
        c1:e0:50:de:7c:60:32:66:a0:12:69:a6:75:5f:88:
        90:0f:7a:73:e3:48:40:af:4e:dd:68:54:e6:34:bc:
```

**Alice:**

```
openssl req -new -config alice.cnf -key alice_private_key.pem -out alice.csr
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl req -new -config alice.cnf -key alice_private_key.pem -out alice.csr
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl req -text -noout -in alice.csr
Certificate Request:
Data:
Version: 1 (0x0)
Subject: CN = Alice1.com, emailAddress = Alice1@gmail.com, O = Alice1, L = Nagpur, C = IN
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (1024 bit)
            Modulus:
                00:a9:1d:9d:4c:ed:6a:9f:fa:71:83:21:96:42:00:
                c3:cc:42:02:b0:8f:d8:24:d0:1d:db:21:d6:55:1c:
                cc:95:2e:9e:9b:a1:06:43:b4:bc:ea:d2:b0:9f:2f:
                aa:9c:b2:5e:a7:8e:27:84:b6:12:ef:0f:a0:7b:fc:
                6d:71:5c:40:06:fc:fc:3b:d9:1a:81:f4:60:ca:0d:
                7c:dc:fc:30:94:6a:2f:34:e1:66:bb:59:bc:b2:4e:
                fc:33:a7:b0:38:d9:da:a6:8e:25:f6:a2:9a:4b:98:
                f6:36:36:35:55:f9:5a:09:20:f9:a9:3b:59:a8:85:
                80:67:7d:bb:bf:bb:0f:94:a1
            Exponent: 65537 (0x10001)
Attributes:
    (none)
Requested Extensions:
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
6b:fc:06:17:10:18:1d:a3:17:f6:21:ea:a1:69:fa:81:aa:93:
b4:ab:65:b4:c1:c3:63:5f:0f:74:00:46:e0:20:cc:8c:84:b4:
2a:86:42:2b:cf:88:00:e8:65:4c:4d:ef:a4:af:f6:87:31:d2:
94:02:51:8d:1d:c9:16:1c:43:f1:8a:15:4b:09:9f:fd:8a:24:
8b:39:82:c4:a7:40:a9:66:fd:16:3c:ab:c2:81:55:bd:d9:e7:
f5:cb:35:6d:5c:10:95:c4:8c:46:86:84:62:2b:28:93:ef:5b:
f1:3f:e9:79:2b:99:d4:3c:0f:0d:2e:12:c2:19:fa:f6:55:d1:
ae:a6
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

**Bob:**

```
openssl req -new -config bob.cnf -key bob_private_key.pem -out bob.csr
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl req -new -config bob.cnf -key bob_private_key.pem -out bob.csr
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl req -text -noout -in bob.csr
Certificate Request:
Data:
Version: 1 (0x0)
Subject: CN = Bob1.com, emailAddress = Bob1@gmail.com, O = Bob1, L = Kolkata, C = IN
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
        Public-Key: (256 bit)
        pub:
            04:56:32:6a:a1:4d:e9:bc:d8:90:bc:f7:08:b3:f6:
            b8:4c:cb:3b:f8:33:b5:03:31:72:6e:e8:2c:36:c6:
            cf:b0:e9:3e:6c:86:5e:e4:c5:83:46:f8:b6:e7:e8:
            3b:c5:f4:c8:0a:88:84:c7:29:ae:fc:49:d7:2a:aa:
            9f:63:d6:13:55
        ASN1 OID: prime256v1
        NIST CURVE: P-256
Attributes:
    (none)
Requested Extensions:
Signature Algorithm: ecdsa-with-SHA256
Signature Value:
30:46:02:21:00:c0:20:cf:b3:5f:90:b9:14:97:59:d6:c0:22:
e0:94:7a:85:e2:c9:7:d6:8c:ff:a5:42:b9:e0:be:41:61:6c:
46:02:21:00:cc:1b:a3:9d:69:cd:e6:05:fb:fc:63:ef:96:5e:
04:f5:4d:d3:fe:49:c6:17:90:bc:1d:9c:15:01:2a:a5:23:5b
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

## **Step 4: Checking the Integrity of received CSR:**

### **Intermediate CA:**

- Creating digest of ‘int.csr’:**

```
openssl req -in int.csr -noout -text | openssl dgst -md5 -out int_csr_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl req -in int.csr -noout -text | openssl dgst -md5 -out int_csr_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat int_csr_digest
MD5(stdin)= 6d5fb2e9809518503b789960fabbdef
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- Siging the digest with ‘int\_private\_key.pem’:**

```
openssl pkeyutl -sign -in int_csr_digest -out int_csr_sign -inkey int_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkeyutl -sign -in int_csr_digest -out int_csr_sign -inkey int_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- Sent ‘int.csr’ and ‘int\_csr\_sign’ to Root CA**

- RootCA creates digest of ‘int.csr’**

```
openssl req -in int.csr -noout -text | openssl dgst -md5 -out int_csr_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl req -in int.csr -noout -text | openssl dgst -md5 -out int_csr_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ ls
int.csr int_csr_digest_generated int_csr_sign root.cnf root.csr root_private_key.pem root.txt
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

- RootCA extracts the public key from ‘int.csr’:**

```
openssl req -in int.csr -pubkey -noout > int_public_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl req -in int.csr -pubkey -noout > int_public_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ cat int_public_key.pem
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEAAQAg8AMIIICgKCAgEAzVqG585G930eyMtVKPG
G9eMR27Jt5YIK0Wgyv9e050rw1hzZeHYKFPXrAIPAJzcC1G7J70me2qqXFLVX3o6
QvVZ7unEZDVko7yvJ0J8X+WoyKiP6GEwT2iTKgAmgiInhiPxmdTFBrkUCPeW0032
okE0eD0YmGPFw2LHc7COV/1YWz8l0i7hdswt8fSbBI4fGrC8No2gtBmjRUFDe4u
4EBex/90NPwwd2Vtv2XzwrXHDt2xjNzkYMvDJzcI5KdveCusKOJ+YBtpymby00
pqjB4FDefGAYzqAsaaZ1X4i0D3pz40hAr07daFRuNLyln9CDX9BRWY3xsKOLOz5d
1BTjXrcJXY0tvvU80+34D5zh24vhvoEwnVye80aQ7KDgGA0pkCx2R0gYuw8pv27
1NbBPko+V18dHH6R6p406h0UtpNE/xqit5c46B7bg5x+ddVpV6URAxon+7qdEshh
nQWi3DR5yqfKLxkhCF1+gxtafm/efLDj1qauMAXNqgUIxwp25U7IZr9HovJU5Rim
tVb0CMxf62q8yF6ufuZLRndoWijJTOnA4WHgKtHaqiiAUzD4EEwMriCx01EZDyP8
YwBV0IBVJ5/qdXQPmqHa5lrByPQfNY5iuvmvE5lARKIyXQewjPNrp0bCcj20PL2A
4UD16YwSmXF/URrfX6qi1HosCAwEAAQ==
-----END PUBLIC KEY-----
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

- RootCA verifying the authenticity of digest (i.e., verifying the authenticity of csr):**

```
openssl pkeyutl -verify -sigfile int_csr_sign -in int_csr_digest_generated -inkey int_public_key.pem -pubin
int_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl pkeyutl -verify -sigfile int_csr_sign -in int_csr_digest_generated -inkey int_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

**Alice:**

- a. **Creating digest of ‘alice.csr’:**

```
openssl req -in alice.csr -noout -text | openssl dgst -md5 -out alice_csr_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl req -in alice.csr -noout -text | openssl dgst -md5 -out alice_csr_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ cat alice_csr_digest
MD5(stdin)= 5fafc1b9b92088293b8b558f26089a12
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

- b. **Signing the digest with ‘alice\_private\_key.pem’:**

```
openssl pkeyutl -sign -in alice_csr_digest -out alice_csr_sign -inkey
alice_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl pkeyutl -sign -in alice_csr_digest -out alice_csr_sign -inkey alice_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ ls
alice.cnf  alice.csr  alice_csr_digest  alice_csr_sign  alice_private_key.pem  alice.txt
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

- c. **Sent ‘alice.csr’ and ‘alice\_csr\_sign’ to Intermediate CA**

- d. **Intermediate CA creates digest of ‘alice.csr’**

```
openssl req -in alice.csr -noout -text | openssl dgst -md5 -out
alice_csr_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl req -in alice.csr -noout -text | openssl dgst -md5 -out alice_csr_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat alice_csr_digest_generated
MD5(stdin)= 5fafc1b9b92088293b8b558f26089a12
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- e. **Intermediate CA extracts the public key from ‘alice.csr’:**

```
openssl req -in alice.csr -pubkey -noout > alice_public_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl req -in alice.csr -pubkey -noout > alice_public_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat alice_public_key.pem
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQAA4GNADCBiOKBg0CpHZ1M7Wqf+nGDIZZCAMPMoQgKw
j9gk0B3bIdZVHMyVLp6boQZDtLzq0rCfL6qcsl6njieEthLvD6B7/G1xEAG/Pw7
2RqBT2DKDzc/DCuai804Wa7WbyyTvwzp7A42dqmjiXZoppLmPY2NjVV+VoJIPmp
O1mohYBnfbu/u4+UoQIDAQAB
-----END PUBLIC KEY-----
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- f. **Intermediate CA verifying the authenticity of digest (i.e., verifying the authenticity of csr):**

```
openssl pkeyutl -verify -sigfile alice_csr_sign -in alice_csr_digest_generated -inkey
alice_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkeyutl -verify -sigfile alice_csr_sign -in alice_csr_digest_generated -inkey alice_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

**Bob:**

- a. **Creating digest of ‘bob.csr’:**

```
openssl req -in bob.csr -noout -text | openssl dgst -md5 -out bob_csr_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl req -in bob.csr -noout -text | openssl dgst -md5 -out bob_csr_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ cat bob_csr_digest
MD5(stdin)= d2b234b5eeb6e918319a6ebc322ae937
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

- b. **Signing the digest with ‘bob\_private\_key.pem’:**

```
openssl pkcs12 -sign -in bob_csr_digest -out bob_csr_sign -inkey bob_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl pkcs12 -sign -in bob_csr_digest -out bob_csr_sign -inkey bob_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ ls
bob.cnf bob.csr bob_csr digest bob_csr_sign bob_private_key.pem bob.txt
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

- c. **Sent ‘bob.csr’ and ‘bob\_csr\_sign’ to Intermediate CA**

- d. **Intermediate CA creates digest of ‘bob.csr’**

```
openssl req -in bob.csr -noout -text | openssl dgst -md5 -out bob_csr_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl req -in bob.csr -noout -text | openssl dgst -md5 -out bob_csr_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat bob_csr_digest_generated
MD5(stdin)= d2b234b5eeb6e918319a6ebc322ae937
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- e. **Intermediate CA extracts the public key from bob.csr’:**

```
openssl req -in bob.csr -pubkey -noout > bob_public_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl req -in bob.csr -pubkey -noout > bob_public_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat bob_public_key.pem
-----BEGIN PUBLIC KEY-----
MFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAEVjJqoU3pvNiQvPcIs/a4TMs7+D01
AzFybugsNsbPs0k+bIZe5MDRvi25+g7xfTICoExymu/EnXKqqfY9YTQ== 
-----END PUBLIC KEY-----
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- f. **Intermediate CA verifying the authenticity of digest (i.e., verifying the authenticity of csr):**

```
openssl pkcs12 -verify -sigfile bob_csr_sign -in bob_csr_digest_generated -inkey bob_public_key.pem -pubin bob_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkcs12 -verify -sigfile bob_csr_sign -in bob_csr_digest_generated -inkey bob_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

## **Step 5: Creating the Certificate (.crt) by signing CSR:**

### **RootCA (self-signed):**

```
openssl x509 -req -days 730 -in root.csr -signkey root_private_key.pem -out root.crt -extfile root.txt
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl x509 -req -days 730 -in root.csr -signkey root_private_key.pem -out root.crt -extfile root.txt
Certificate request self-signature ok
subject=CN = iTS ROOT R1, emailAddress = rootCA@gmail.com, O = iTS ROOT R1, L = Hyderabad, C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl x509 -text -noout -in root.crt
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
13:76:03:b7:4b:f7:89:17:1f:4d:f9:ca:4f:ab:32:d2:7b:c2:ae:be
Signature Algorithm: ecdsa-with-SHA256
Issuer: CN = iTS ROOT R1, emailAddress = rootCA@gmail.com, O = iTS ROOT R1, L = Hyderabad, C = IN
Validity
Not Before: Mar 15 12:09:54 2024 GMT
Not After : Mar 15 12:09:54 2026 GMT
Subject: CN = iTS ROOT R1, emailAddress = rootCA@gmail.com, O = iTS ROOT R1, L = Hyderabad, C = IN
Subject Public Key Info:
Public Key Algorithm: id-ecPublicKey
Public-Key: (512 bit)
pub:
04:2f:04:ea:e9:61:ae:29:35:9f:8f:14:64:5b:9a:
3c:49:19:f2:e5:8b:a9:fe:60:bf:46:af:66:72:49:
89:46:44:49:af:49:0f:bl:le:25:33:lc:87:9b:0e:
c5:c0:46:02:23:8f:3a:51:ab:96:1c:8d:dl:ef:9f:
3c:al:6b:08:79:a1:db:17:11:57:24:27:fb:50:bb:
30:a2:61:5c:2f:3c:4c:0:f8:c2:36:71:3f:61:64:
38:9c:4c:a3:57:a3:d8:fb:3a:8c:f3:ae:db:85:f1:
24:7f:e9:41:a1:f0:ee:1b:31:fd:63:fo:bf:6a:d5:
dc:17:87:a0:af:9d:81:45:7c
ASN1 OID: brainpoolP512r1
X509v3 extensions:
X509v3 Authority Key Identifier:
Keyid:5D:E8:30:4B:C1:75:E1:65:7A:74:B4:E2:70:2E:52:CA:5C:83:41:1D
DirName:/CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN
serial:13:76:03:B7:4B:F7:89:17:1F:4D:F9:CA:4F:AB:32:D2:7B:C2:AE:BE
X509v3 Basic Constraints:
CA:TRUE
X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
X509v3 Subject Key Identifier:
5D:E8:30:4B:C1:75:E1:65:7A:74:B4:E2:70:2E:52:CA:5C:83:41:1D
Signature Algorithm: ecdsa-with-SHA256
Signature Value:
30:81:48:02:40:76:08:ce:1c:18:29:5a:8a:75:0a:c2:f8:fc:
41:ed:19:20:8d:20:42:6e:db:42:f0:f2:ae:54:ef:98:e2:75:
ba:3a:03:ca:c3:b0:4a:f1:3f:cfa5:6e:83:17:f9:11:4e:68:
f5:19:b7:78:36:cb:45:77:85:e7:c3:e9:15:0:3a:02:40:32:
6f:f3:ee:88:76:2c:6a:7f:04:77:3d:8c:d4:f0:fe:20:22:1a:
35:94:57:e8:59:65:59:71:37:a1:11:0e:6f:46:3b:be:65:cf:
45:10:id:f1:e3:a8:fc:f1:f5:3a:08:7a:2e:33:48:3e:2e:bc:
d1:57:7e:8d:3a:6a:3a:38:48
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

### **IntermediateCA (signed by RootCA):**

```
openssl x509 -req -in int.csr -CA root.crt -extfile int.txt -CAkey root_private_key.pem
-CAcreateserial -out int.crt -days 730 -sha256
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl x509 -req -in int.csr -CA root.crt -extfile int.txt -CAkey root_private_key.pem -CAcreateserial -out int.crt -days 730 -sha256
Certificate request self-signature ok
subject=CN = iTS CA IR3, emailAddress = intCA@gmail.com, O = iTS CA IR3, L = Nashik, C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl x509 -text -noout -in int.crt
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
75:04:75:07:53:3b:89:b4:fc:ea:9f:5a:e9:f5:c4:34:41:08:f1:15
Signature Algorithm: ecdsa-with-SHA256
Issuer: CN = iTS ROOT R1, emailAddress = rootCA@gmail.com, O = iTS ROOT R1, L = Hyderabad, C = IN
Validity
Not Before: Mar 15 12:11:25 2024 GMT
Not After : Mar 15 12:11:25 2026 GMT
Subject: CN = iTS CA IR3, emailAddress = intCA@gmail.com, O = iTS CA IR3, L = Nashik, C = IN
Subject Public Key Info:
Public Key Algorithm: rsaEncryption
Public-Key: (4096 bit)
Modulus:
00:cd:5a:86:4b:ce:46:f7:74:1e:c8:cb:55:26:43:
c6:1b:d7:8c:47:6e:c9:b7:96:08:2b:45:a0:cb:0f:
5e:43:94:2b:c3:58:73:65:e1:d8:28:53:07:ac:02:
5e:43:94:2b:c3:58:73:65:e1:d8:28:53:07:ac:02:
5e:43:94:2b:c3:58:73:65:e1:d8:28:53:07:ac:02:
5e:43:94:2b:c3:58:73:65:e1:d8:28:53:07:ac:02:
5e:43:94:2b:c3:58:73:65:e1:d8:28:53:07:ac:02:
```

### Alice (signed by IntermediateCA):

```
openssl x509 -req -in alice.csr -CA int.crt -extfile alice.txt -CAkey int_private_key.pem  
-CAcreateserial -out alice.crt -days 730 -sha256
```

```
mayuresh@mayuresh-HP-Laptop:/Desktop/secure_chat_submission/int$ openssl x509 -req -in alice.csr -CA int.crt -extfile alice.txt -CAkey int_private_key.pem -CAcreateserial -out alice.crt -days 730 -sha256  
Certificate request self-signature ok  
subject=CN = Alicel.com, emailAddress = Alice1@gmail.com, O = Alicel, L = Nagpur, C = IN  
mayuresh@mayuresh-HP-Laptop:/Desktop/secure_chat_submission/int$ openssl x509 -text -noout -in alice.crt  
Certificate:  
Data:  
    Version: 3 (0x2)  
    Serial Number:  
        40:fb:0c:9b:f7:0f:99:a5:f2:5c:07:04:89:4e:bd:51:a0:17:1f  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: CN = iTS CA IR3, emailAddress = intCA@gmail.com, O = iTS CA IR3, L = Nashik, C = IN  
    Validity  
        Not Before: Mar 15 12:16:01 2024 GMT  
        Not After : Mar 15 12:16:01 2026 GMT  
    Subject: CN = Alicel.com, emailAddress = Alice1@gmail.com, O = Alicel, L = Nagpur, C = IN  
    Subject Public Key Info:  
        Public Key Algorithm: rsaEncryption  
            Public-Key: (1024 bit)  
                Modulus:  
                    00:a9:1d:d0:4c:ed:6a:9f:fa:71:83:21:96:42:00:  
                    c3:cc:42:02:00:8f:0d:00:00:00:00:00:00:00:00:  
                    cc:00:00:00:00:00:00:00:00:00:00:00:00:00:00:  
                    a8:9c:b2:5e:a7:8e:00:71:84:b6:12:ef:0f:a0:71:f3  
                    0d:71:5c:40:06:fc:fc:3b:d9:1a:91:4f:60:c3:0d:  
                    7c:dc:fc:30:94:6a:2f:34:e1:66:bb:59:bc:b2:4e:  
                    fc:33:a7:0b:38:d9:da:a6:8e:25:f6:n2:9a:9b:  
                    f6:36:3e:35:55:f9:5a:09:20:f9:a9:3b:59:a8:85:  
                    80:67:7d:bb:bf:bb:8f:94:a1
```

### Bob (signed by IntermediateCA):

```
openssl x509 -req -in bob.csr -CA int.crt -extfile bob.txt -CAkey int_private_key.pem  
-CAcreateserial -out bob.crt -days 730 -sha256
```

```
mayuresh@mayuresh-HP-Laptop:/Desktop/secure_chat_submission/int$ openssl x509 -req -in bob.csr -CA int.crt -extfile bob.txt -CAkey int_private_key.pem -CAcreateserial -out bob.crt -days 730 -sha256  
Certificate request self-signature ok  
subject=CN = Bob1.com, emailAddress = Bob1@gmail.com, O = Bob1, L = Kolkata, C = IN  
mayuresh@mayuresh-HP-Laptop:/Desktop/secure_chat_submission/int$ openssl x509 -text -noout -in bob.crt  
Certificate:  
Data:  
    Version: 3 (0x2)  
    Serial Number:  
        47:4d:db:73:b3:57:12:2b:1f:22:c2:fb:11:2c:ac:f4:80:2c:01:4d  
    Signature Algorithm: sha256WithRSAEncryption  
    Issuer: CN = iTS CA IR3, emailAddress = intCA@gmail.com, O = iTS CA IR3, L = Nashik, C = IN  
    Validity  
        Not Before: Mar 15 12:17:16 2024 GMT  
        Not After : Mar 15 12:17:16 2026 GMT  
    Subject: CN = Bob1.com, emailAddress = Bob1@gmail.com, O = Bob1, L = Kolkata, C = IN  
    Subject Public Key Info:  
        Public Key Algorithm: id-ecPublicKey  
            Public-Key: (256 bit)  
                pub:  
                    04:56:32:6a:a1:4d:e9:bc:d8:90:bc:f7:08:b3:f6:  
                    b8:4c:cb:3b:f8:33:b5:03:31:72:6e:e8:2c:36:c6:  
                    cf:bd:e9:3e:6c:86:5e:e4:c5:83:46:f8:b6:e7:e8:  
                    3b:c5:f4:c8:0a:88:84:c7:29:ae:fc:49:d7:2a:aa:  
                    9f:63:06:13:55  
                ASN1 OID: prime256v1  
                NIST CURVE: P-256  
X509v3 extensions:  
    X509v3 Authority Key Identifier:  
        75:43:D6:5C:11:5F:47:AF:92:87:D0:2B:DA:9B:E3:DF:CF:81:54:6B  
    X509v3 Basic Constraints:  
        CA:FALSE  
    X509v3 Key Usage:  
        Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment  
    X509v3 Subject Key Identifier:  
        BE:57:0B:B9:84:DB:08:34:12:78:46:5F:00:1F:FE:33:0D:2D:06:91  
Signature Algorithm: sha256WithRSAEncryption
```

## **Step 6: Checking the Integrity of received CRT:**

### **Intermediate CA:**

- RootCA creating digest of ‘int.crt’:**

```
openssl x509 -in int.crt -outform DER | openssl dgst -md5 > int_crt_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl x509 -in int.crt -outform DER | openssl dgst -md5 > int_crt_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ cat int_crt_digest
MD5(stdin)= 6d7038f80387961e7a26b76cd8c6dc0a
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

- RootCA signing the digest with ‘root\_private\_key.pem’:**

```
openssl pkeyutl -sign -in int_crt_digest -out int_crt_sign -inkey root_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl pkeyutl -sign -in int_crt_digest -out int_crt_sign -inkey root_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ ls
int.crt int_crt_digest int_crt_sign int.csr int_csr_digest_generated int_csr_sign int_public_key.pem int.txt root.cnf root.crt root.csr root_private_key.pem root.txt
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$
```

- RootCA sent ‘int.crt’ and ‘int\_crt\_sign’ to Intermediate CA**

- Intermediate CA creates digest of ‘int.crt’**

```
openssl x509 -in int.crt -outform DER | openssl dgst -md5 > int_crt_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl x509 -in int.crt -outform DER | openssl dgst -md5 > int_crt_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat int_crt_digest_generated
MD5(stdin)= 6d7038f80387961e7a26b76cd8c6dc0a
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- RootCA’s public key is known to Intermediate CA**

- Verifying the authenticity of the certificate:**

```
openssl pkeyutl -verify -sigfile int_crt_sign -in int_crt_digest_generated -inkey root_public_key.pem -pubin
root_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkeyutl -verify -sigfile int_crt_sign -in int_crt_digest_generated -inkey root_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

### **Alice:**

- Intermediate CA creating digest of ‘alice.crt’:**

```
openssl x509 -in alice.crt -outform DER | openssl dgst -md5 > alice_crt_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl x509 -in alice.crt -outform DER | openssl dgst -md5 > alice_crt_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat alice_crt_digest
MD5(stdin)= 3456432a0c9f05ed097eb8b61c9de164
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- Intermediate CA signing the digest with ‘int\_private\_key.pem’:**

```
openssl pkeyutl -sign -in alice_crt_digest -out alice_crt_sign -inkey int_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkeyutl -sign -in alice_crt_digest -out alice_crt_sign -inkey int_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ ls | grep int_private_key.pem
int_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

- Intermediate CA sent ‘alice.crt’ and ‘alice\_crt\_sign’ to Alice**

- Alice creates digest of ‘alice.crt’:**

```
openssl x509 -in alice.crt -outform DER | openssl dgst -md5 >
alice_crt_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl x509 -in alice.crt -outform DER | openssl dgst -md5 > alice_crt_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ cat alice_crt_digest_generated
MD5(stdin)= 3456432a0c9f05ed097eb8b61c9de164
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

e. Intermediate CA's public key is known to Alice

f. Verifying the authenticity of the certificate:

```
openssl pkeyutl -verify -sigfile alice_crt_sign -in alice_crt_digest_generated -inkey int_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl pkeyutl -verify -sigfile alice_crt_sign -in alice_crt_digest_generated -inkey int_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$
```

**Bob:**

a. Intermediate CA creating digest of 'bob.crt':

```
openssl x509 -in bob.crt -outform DER | openssl dgst -md5 > bob_crt_digest
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl x509 -in bob.crt -outform DER | openssl dgst -md5 > bob_crt_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ cat bob_crt_digest
MD5(stdin)= 29428a6bcc7d6ddfb57bc69f71ebd21
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$
```

b. Intermediate CA signing the digest with 'int\_private\_key.pem':

```
openssl pkeyutl -sign -in bob_crt_digest -out bob_crt_sign -inkey int_private_key.pem
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl pkeyutl -sign -in bob_crt_digest -out bob_crt_sign -inkey int_private_key.pem
```

c. Intermediate CA sent 'bob.crt' and 'bob\_crt\_sign' to Alice

d. Bob creates digest of 'bob.crt':

```
openssl x509 -in bob.crt -outform DER | openssl dgst -md5 > bob_crt_digest_generated
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl x509 -in bob.crt -outform DER | openssl dgst -md5 > bob_crt_digest_generated
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ cat bob_crt_digest_generated
MD5(stdin)= 29428a6bcc7d6ddfb57bc69f71ebd21
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

e. Intermediate CA's public key is known to Alice

f. Verifying the authenticity of the certificate:

```
openssl pkeyutl -verify -sigfile bob_crt_sign -in bob_crt_digest_generated -inkey int_public_key.pem -pubin
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl pkeyutl -verify -sigfile bob_crt_sign -in bob_crt_digest_generated -inkey int_public_key.pem -pubin
Signature Verified Successfully
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$
```

## **Step 7: Verifying the Certificate Chain for RootCA, InterCA, Alice and Bob**

### **RootCA:**

```
openssl verify -CAfile root.crt root.crt
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ openssl verify -CAfile root.crt root.crt
root.crt: OK
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/root$ █
```

### **IntermediateCA:**

```
openssl verify -CAfile root.crt int.crt
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ openssl verify -CAfile root.crt int.crt
int.crt: OK
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/int$ █
```

### **Alice:**

```
openssl verify -CAfile root.crt -untrusted int.crt alice.crt
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ openssl verify -CAfile root.crt -untrusted int.crt alice.crt
alice.crt: OK
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/alice$ █
```

### **Bob:**

```
openssl verify -CAfile root.crt -untrusted int.crt bob.crt
```

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ openssl verify -CAfile root.crt -untrusted int.crt bob.crt
bob.crt: OK
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission/bob$ █
```

---

## **TASK 2: Secure Chat App**

### **Secure Chat Output:**

The screenshot shows two terminal windows side-by-side, both titled "Secure Chat Output".

**Terminal 1 (Left):**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -s
Server Info:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for connection....
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
Msg recv from Client: chat_START_SSL
Msg sent to Client: chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired

Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired

Depth:0, Cert Info: /CN=Alicel.com/emailAddress=Alice1@gmail.com/O=Alice1/L=Nagpur/C=IN
Certificate having depth 0 is NOT expired

.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
-----
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****
From Client: Hi Bob1
Enter message for Client: Hi Alice1

From Client: The weather seems good today!
Enter message for Client: Yes. Can we go out for picnic?

From Client: Sure.
Enter message for Client: Tomorrow 8 am is fine for you?

From Client: Yes, lets go by car.
Enter message for Client: ok

From Client: See you tomorrow
Enter message for Client: bye

From Client: chat close
***** Chat closed *****

-----
Server socket created successfully
-----
Server started listening for connection....
```

**Terminal 2 (Right):**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -c mayuresh-HP-Laptop
Client will chat with server:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Client Info:
Client IP: 127.0.1.1
Client Port: 6001
-----
Client socket created Successfully
-----
Msg sent to Server: chat_hello
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired

Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired

Depth:0, Cert Info: /CN=Bob1.com/emailAddress=Bob1@gmail.com/O=Bob1/L=Kolkata/C=IN
Certificate having depth 0 is NOT expired

.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
-----
Client connected to Server using DTLS
-----
***** Client has started chatting with Server *****
Enter message for Server: Hi Bob1

From Server: Hi Alice1
Enter message for Server: The weather seems good today!

From Server: Yes. Can we go out for picnic?
Enter message for Server: Sure.

From Server: Tomorrow 8 am is fine for you?
Enter message for Server: Yes, lets go by car.

From Server: ok
Enter message for Server: See you tomorrow

From Server: bye
Enter message for Server: chat_close

***** Chat closed *****
```

At the bottom of Terminal 2, it shows the prompt "mayuresh@mayuresh-HP-Laptop:~/Desktop/secure\_chat\_submission\$".

## Flow diagram depicting DTLS 1.2 message flow:



## Wireshark Trace of entire chat:

The screenshot shows a Wireshark capture of a secure chat session. The packet list pane shows 41 captured frames, mostly TLS handshakes and application data. The details pane shows the structure of a TLS message, and the bytes pane shows the raw hex and ASCII data.

Selected packet details:

- Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits) on interface lo, id 0
- Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1
- User Datagram Protocol, Src Port: 6001, Dst Port: 5001
- Data (10 bytes)  
Data: 636861745f68656c6c6f  
[Length: 10]

Selected packet bytes:

0000	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00	.....	E
0010	00 26 35 66 40 00 40 11 05 5f 7f 00 01 01 7f 00	&5f@ @ .....	
0020	01 01 17 71 13 89 00 12 00 26 63 68 61 74 5f 68	...g ..&chat_h	
0030	65 6c 6c 6f	ello	

### a) Running the secure chat application:

We start the server using the below command:

In the screenshot below, Bob is the server running on **Port:5001**.

```
$ secure_chat_app -s
```

We start the client using the below command:

In the screenshot below, Alice is the client running on **Port:6001**.

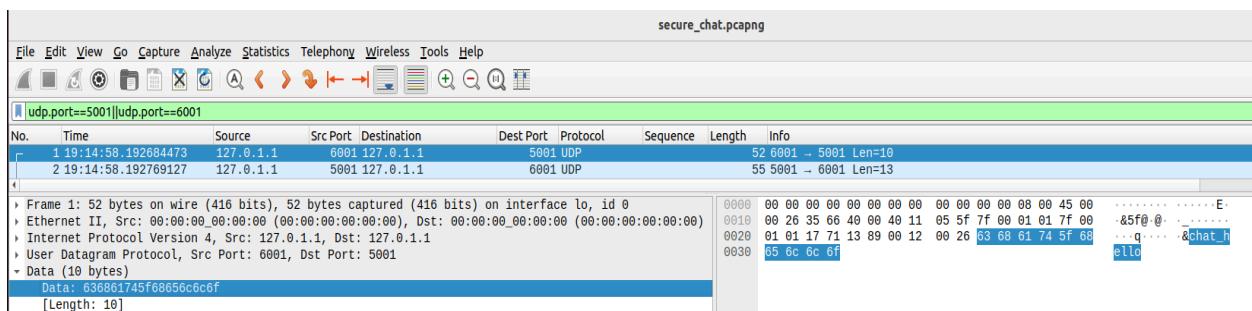
```
$ secure_chat_app -c <server_hostname>
```

### b) Exchanging the ‘hello control messages’:

We can observe that these messages are exchanged in plain text.

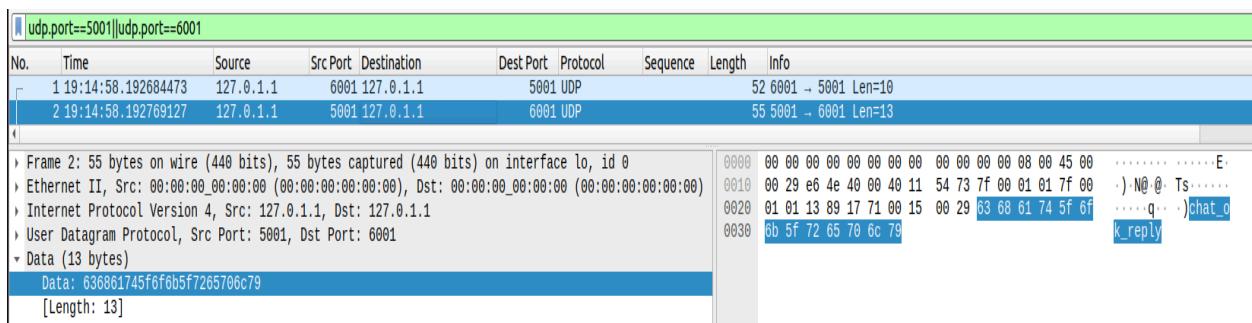
Messages sent by Client to Server:

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -c mayuresh-HP-Laptop
Client will chat with server:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Client Info:
Client IP: 127.0.1.1
Client Port: 6001
-----
Client socket created Successfully
-----
Msg sent to Server: chat_hello
Msg recv from Server: chat ok reply
```



Messages sent by Server to the Client:

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -s
Server Info:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Server socket created successfully
-----
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
```



### c) Exchanging the ‘control messages’:

We can observe that these messages are exchanged in plain text.

## **Messages sent by Client to Server:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -c mayuresh-HP-Laptop
Client will chat with server:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Client Info:
Client IP: 127.0.1.1
Client Port: 6001
-----
Client socket created Successfully
-----
Msg sent to Server: chat hello
Msg recv from Server: chat ok reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_ACK
```

udp.port==5001  udp.port==6001										
No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info	
1	19:14:58.192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	6001 → 5001 Len=10		
2	19:14:58.192769127	127.0.1.1	5001	127.0.1.1	6001	UDP	55	5001 → 6001 Len=13		
3	19:14:58.192822848	127.0.1.1	6001	127.0.1.1	5001	UDP	56	6001 → 5001 Len=14		
4	19:14:58.192885561	127.0.1.1	5001	127.0.1.1	6001	UDP	60	5001 → 6001 Len=18		

#### **Messages sent by Server to the Client:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -s
Server Info:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Server socket created successfully
-----
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
Msg recv from Client: chat_START_SSL
Msg sent to Client: chat_START_SSL_ACK
```

udp.port==5001  udp.port==6001									
No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info
1	19:14:58.192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	6001 → 5001 Len=10	
2	19:14:58.192769127	127.0.1.1	5001	127.0.1.1	6001	UDP	55	5001 → 6001 Len=13	
3	19:14:58.19282848	127.0.1.1	6001	127.0.1.1	5001	UDP	56	6001 → 5001 Len=14	
4	19:14:58.192885561	127.0.1.1	5001	127.0.1.1	6001	UDP	60	5001 → 6001 Len=18	

Frame 4: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface lo, id 0

Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)

Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1

User Datagram Protocol, Src Port: 5001, Dst Port: 6001

Data (18 bytes)

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 45 00 . . . .
0010 00 2e e6 4f 40 00 40 11 54 6d 7f 00 01 01 7f 00 ..0@.Tm. .
0020 01 01 13 89 17 71 00 1a 00 2e 53 68 61 74 5f 53 ..q..chat. .
0030 54 41 52 54 5f 53 53 4c 5f 41 43 4b TART_SSL_ACK
```

Data: 636661745f53544152545f53534c5f41434b

[Length: 18]

### c-i) Client sends 'Client Hello' message to server:

When the client receives 'chat\_START\_SSL\_ACK' from the server, it initiates the DTLS 1.2 handshake by sending a 'Client Hello' message to Server.

No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info
1	19:14:58.192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	6001	- 5001 Len=10
2	19:14:58.192769127	127.0.1.1	5001	127.0.1.1	6001	UDP	55	5001	- 6001 Len=13
3	19:14:58.192822848	127.0.1.1	6001	127.0.1.1	5001	UDP	56	6001	- 5001 Len=14
4	19:14:58.192865561	127.0.1.1	5001	127.0.1.1	6001	UDP	60	5001	- 6001 Len=18
5	19:14:58.198354678	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	217		Client Hello
6	19:14:58.199086081	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	76		Hello Verify Request
7	19:14:58.199177008	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	223		Client Hello
8	19:14:58.201719897	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	3351		Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done

Frame 5: 217 bytes on wire (1736 bits), 217 bytes captured (1736 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Client Hello  
 Content Type: Handshake (22)  
 Version: DTLS 1.2 (0xfefd)  
 Epoch: 0  
 Sequence Number: 0  
 Length: 162  
   Handshake Protocol: Client Hello  
 Handshake Type: Client Hello (1)  
 Length: 150  
 Message Sequence: 0  
 Fragment Offset: 0  
 Fragment Length: 150  
 Version: DTLS 1.2 (0xfefd)  
 Random: 76fb99619277c6b336c47fcc4fe64c257d3500f6b0c31369628f9edfed440c

Frame 6: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 5001, Dst Port: 6001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Client Hello  
 Content Type: Handshake (22)  
 Version: DTLS 1.2 (0xfefd)  
 Epoch: 0  
 Sequence Number: 0  
 Length: 162  
   Handshake Protocol: Client Hello  
 Handshake Type: Client Hello (1)  
 Length: 150  
 Message Sequence: 0  
 Fragment Offset: 0  
 Fragment Length: 150  
 Version: DTLS 1.2 (0xfefd)  
 Random: 0b080408050806040105010601030303

Frame 7: 223 bytes on wire (1784 bits), 223 bytes captured (1784 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Client Hello  
 Content Type: Handshake (22)  
 Version: DTLS 1.2 (0xfefd)  
 Epoch: 0  
 Sequence Number: 0  
 Length: 162  
   Handshake Protocol: Client Hello  
 Handshake Type: Client Hello (1)  
 Length: 150  
 Message Sequence: 0  
 Fragment Offset: 0  
 Fragment Length: 150  
 Version: DTLS 1.2 (0xfefd)  
 Random: 010302040205020602

### c-ii) Certificate Exchange and Verification:

#### Certificate Exchange:

- Server replies to Client by a 'Server Hello' message which contains Server's Digital Certificate.
- This certificate contains the server's public key.
- We can observe that, in the 'Server Hello' message, the server also requests for the Client's digital certificate.

#### Server sends its certificate to Client:

No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info
1	19:14:58.192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	6001	- 5001 Len=10
2	19:14:58.192769127	127.0.1.1	5001	127.0.1.1	6001	UDP	55	5001	- 6001 Len=13
3	19:14:58.192822848	127.0.1.1	6001	127.0.1.1	5001	UDP	56	6001	- 5001 Len=14
4	19:14:58.192865561	127.0.1.1	5001	127.0.1.1	6001	UDP	60	5001	- 6001 Len=18
5	19:14:58.198354678	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	217		Client Hello
6	19:14:58.199086081	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	76		Hello Verify Request
7	19:14:58.199177008	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	223		Client Hello
8	19:14:58.201719897	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	3351		Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
9	19:14:58.209198864	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	3656		Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
10	19:14:58.213615221	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	1524		New Session Ticket, Change Cipher Spec, Encrypted Handshake Message

Frame 8: 3351 bytes on wire (26808 bits), 3351 bytes captured (26808 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Server Hello  
 Content Type: Handshake (22)  
 Version: DTLS 1.2 (0xfefd)  
 Epoch: 0  
 Sequence Number: 2  
 Length: 2975  
   Handshake Protocol: Certificate  
 Handshake Type: Certificate (11)  
 Length: 2963  
 Message Sequence: 2  
 Fragment Offset: 0  
 Fragment Length: 2963  
 Certificates Length: 2960  
   Certificates (2960 bytes)  
 Certificate Length: 1001  
 Certificate [truncated]: 3088203e5308201cda00382010202144744db73b357122b1f22c2fb112cacf480...  
 Certificate Length: 1097  
 Certificate [truncated]: 308820445308203a9a003820102021475047507533b89b4fceaf9f5ae9f5c43441...  
 Certificate Length: 853  
 Certificate [truncated]: 308820351308202b6a08038201020214137603b74bf789171f4df9ca4fab32d27b...  
   DTLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange  
   DTLSv1.2 Record Layer: Handshake Protocol: Certificate Request  
   DTLSv1.2 Record Layer: Handshake Protocol: Server Hello Done

Frame 9: 3656 bytes on wire (29280 bits), 3656 bytes captured (29280 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 5001, Dst Port: 6001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Certificate  
 Content Type: Certificate (11)  
 Length: 2963  
 Message Sequence: 2  
 Fragment Offset: 0  
 Fragment Length: 2963  
 Certificates Length: 2960  
   Certificates (2960 bytes)  
 Certificate Length: 1001  
 Certificate [truncated]: 3088203e5308201cda00382010202144744db73b357122b1f22c2fb112cacf480...  
 Certificate Length: 1097  
 Certificate [truncated]: 308820445308203a9a003820102021475047507533b89b4fceaf9f5ae9f5c43441...  
 Certificate Length: 853  
 Certificate [truncated]: 308820351308202b6a08038201020214137603b74bf789171f4df9ca4fab32d27b...  
   DTLSv1.2 Record Layer: Handshake Protocol: Certificate Verify  
   DTLSv1.2 Record Layer: Handshake Protocol: Change Cipher Spec  
   DTLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

Frame 10: 1524 bytes on wire (12192 bits), 1524 bytes captured (12192 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
 Datagram Transport Layer Security  
   DTLSv1.2 Record Layer: Handshake Protocol: Change Cipher Spec  
 Content Type: Change Cipher Spec (1)  
 Length: 1  
 Message Sequence: 3  
 Fragment Offset: 0  
 Fragment Length: 1  
   DTLSv1.2 Record Layer: Handshake Protocol: Encrypted Handshake Message

**Client sends its certificate to Server:** Client provides its certificate to Server in subsequent message.

No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info
1	19:14:58, 192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	5001 - 5001 Len=10	
2	19:14:58, 192769127	127.0.1.1	6001	127.0.1.1	6001	UDP	55	5001 - 6001 Len=13	
3	19:14:58, 192822948	127.0.1.1	6001	127.0.1.1	5001	UDP	56	5001 - 5001 Len=14	
4	19:14:58, 192885561	127.0.1.1	6001	127.0.1.1	6001	UDP	60	5001 - 6001 Len=18	
5	19:14:58, 198354678	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	217	Client Hello	
6	19:14:58, 199866081	127.0.1.1	6001	127.0.1.1	6001	DTLSv1.2	76	76 Hello Verify Request	
7	19:14:58, 199177098	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	223	Client Hello	
8	19:14:58, 201719897	127.0.1.1	6001	127.0.1.1	6001	DTLSv1.2	3351	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done	
9	19:14:58, 209198864	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	3656	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message	
10	19:14:58, 213615221	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	1524	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message	

Frame 9: 3656 bytes on wire (29248 bits), 3656 bytes captured (29248 bits) on interface lo, id 0  
 Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1  
 User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
 Datagram Transport Layer Security  
 - DTLSv1.2 Record Layer: Handshake Protocol: Certificate  
   Content Type: Handshake (22)  
   Version: DTLS 1.2 (0xfeed)  
   Epoch: 0  
   Sequence Number: 2  
   Length: 3183  
 - Handshake Protocol: Certificate  
   Handshake Type: Certificate (11)  
   Length: 3171  
   Message Sequence: 2  
   Fragment Offset: 0  
   Fragment Length: 3171  
   Certificates Length: 3168  
   Certificates (3168 bytes)  
     Certificate Length: 1209  
     > Certificate [truncated]: 3008204b53082029da0030201020214306cb132d3df2fb9728d06ea0a29ac4d9b81  
     Certificate Length: 1097  
     > Certificate [truncated]: 300820445308203a9a003020102021475047507533b89bfceaf0f9ae9f5c4344108  
     Certificate Length: 853  
     > Certificate [truncated]: 300820351308202b6a0030201020214137603b74bf789171f4df9ca4fab32d27bc2  
 - DTLSv1.2 Record Layer: Handshake Protocol: Client Key Exchange  
 - DTLSv1.2 Record Layer: Handshake Protocol: Certificate Verify  
 - DTLSv1.2 Record Layer: Change Cipher Spec Protocol: Change Cipher Spec  
 - Record Layer

## Certificate Verification:

The C code used for 'Certificate Verification' is as below:

```
// Function to perform certificate expiry date checking
int verify_certificate_expiry_date(X509 *cert)
{
    ASN1_TIME *not_after = X509_get_notAfter(cert);
    time_t now = time(NULL);
    struct tm tm_val;
    ASN1_TIME_to_tm(not_after, &tm_val);
    time_t expiry_time = mktime(&tm_val);
    return (now > expiry_time) ? 0 : 1;
}

// This function is called for each certificate in the 'Trust Chain'
int verify_certificate_callback(int preverify_ok, X509_STORE_CTX *ctx)
{
    char buf[256];
    X509 *err_cert;
    int err, depth;

    // Extracting the certificate from SSL/DTLS context
```

```

    err_cert = X509_STORE_CTX_get_current_cert(ctx);
    err = X509_STORE_CTX_get_error(ctx);
    depth = X509_STORE_CTX_get_error_depth(ctx);

    // Printing the depth and Subject name of certificate
    X509_NAME_oneline(X509_get_subject_name(err_cert), buf, 256);
    printf("Depth:%d, Cert Info: %s\n", depth, buf);
    if (!preverify_ok) {
        printf("Certificate Verification Error: num=%d:%s:%s\n", err,
X509_verify_cert_error_string(err), buf);
        exit(1);
    }
    if (preverify_ok) {
        // Verifying the expiry date of certificate
        if (verify_certificate_expiry_date(err_cert)) {
            printf("Certificate having depth %d is NOT expired\n\n",
depth);
        } else {
            printf("Error: Certificate is Expired\n");
            exit(1);
        }
    }
    return preverify_ok;
}

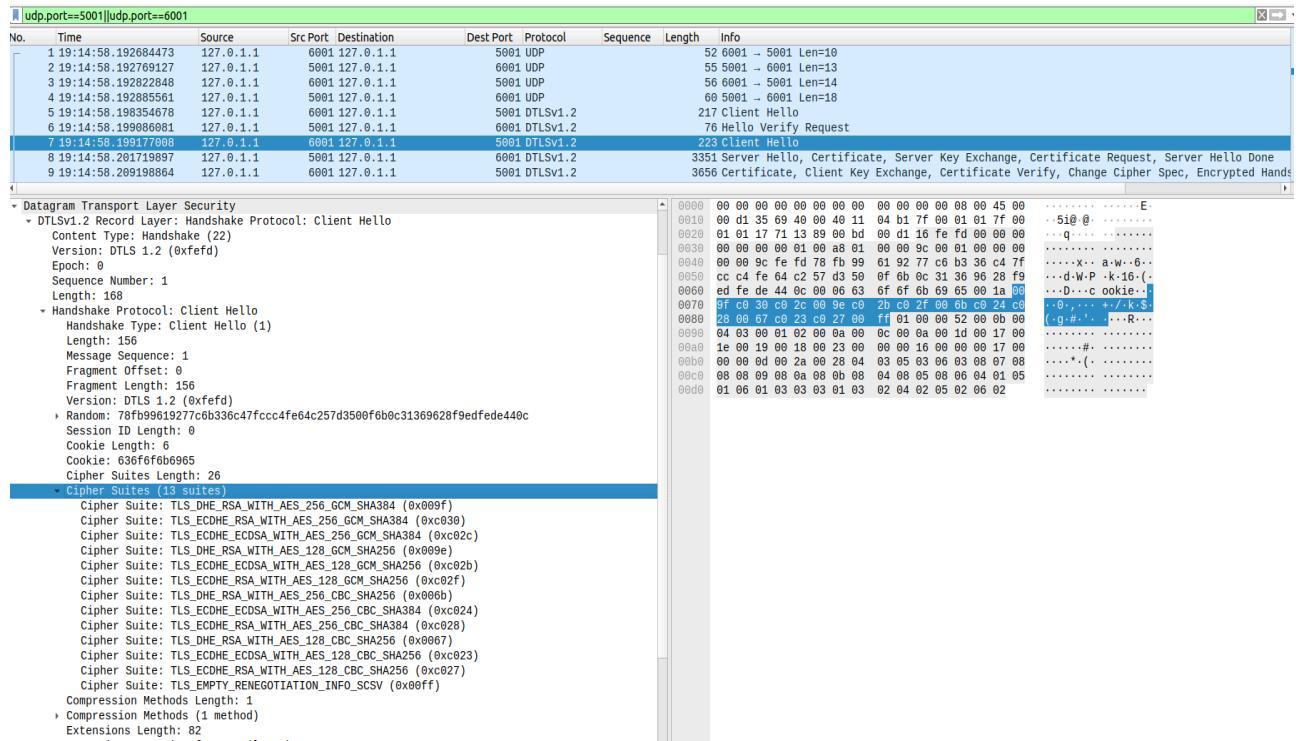
```

1. verify\_certificate\_callback() function is called to verify each certificate in the ‘Trust Chain’.
2. It extracts the digital certificate from the SSL/TLS context object.
3. It prints the certificate subject name and its depth in the trust chain.
4. It verifies the expiry date of the certificate.
5. If the certificate is expired then it throws an error message and stops the DTLS 1.2 handshake.

### c-iii) Cipher Suite Negotiation:

- Client advertises a list of PFS (Perfect Forward Secrecy) supported cipher suites to Server through the ‘Client Hello’ message.
- Server selects one cipher from this list and conveys it to the Client using ‘Server Hello’ message.
- We can observe the same thing in the screenshot below.

## Client sends list of PFS supported cipher suites to Server using ‘Client Hello’ message:



## Part of the code used for sending the PFS supported Cipher suites list to Server through ‘Client Hello’:

```
SSL_CTX *create_client_context() // alice
{
    // Creating context and filling certs and private key in it
    SSL_CTX *ctx = SSL_CTX_new(DTLSv1_2_client_method());

    // Setting the cipher suites list of "12 Cipher Suites that support PFS"
    SSL_CTX_set_cipher_list(ctx,
"DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-
GCM-SHA384:DHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-R-
SA-AES128-GCM-SHA256:DHE-RSA-AES256-SHA256:ECDHE-ECDSA-AES256-SHA384:ECDHE-
-RSA-AES256-SHA384:DHE-RSA-AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-R-
SA-AES128-SHA256");
```

This is Client side code, which sets the list of PFS supported cipher suites in the DTLS 1.2 context of client.

**Server coveys the cipher suite to Client through ‘Server Hello’ message:**

[ udp.port==5001 ] & [ udp.port==6001 ]										
No.	Time	Source	Src Port	Destination	Dest Port	Protocol	Sequence	Length	Info	
1	19:14:58.192684473	127.0.1.1	6001	127.0.1.1	5001	UDP	52	6001 - 5001 Len=10		
2	19:14:58.192769127	127.0.1.1	5001	127.0.1.1	6001	UDP	55	5001 - 6001 Len=13		
3	19:14:58.192822848	127.0.1.1	6001	127.0.1.1	5001	UDP	56	6001 - 5001 Len=14		
4	19:14:58.192885561	127.0.1.1	5001	127.0.1.1	6001	UDP	60	5001 - 6001 Len=18		
5	19:14:58.198354678	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	217	Client Hello		
6	19:14:58.199086081	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	76	Hello Verify Request		
7	19:14:58.199177008	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	223	Client Hello		
8	19:14:58.201719897	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	3351	Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done		
9	19:14:58.201989864	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	3656	Certificate, Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake		
Frame # 3351 bytes on wire (268088 bits), 3351 bytes captured (268088 bits) on interface lo, id 0										
Ethernet II, Src: Intel PRO/100 MT [00:0c:29:b8:bd:5f] (00:0c:29:b8:bd:5f), Dst: Intel PRO/100 MT [00:0c:29:b8:bd:5f] (00:0c:29:b8:bd:5f)										
Internet Protocol Version 4, Src: 127.0.1.1, Dst: 127.0.1.1										
User Datagram Protocol, Src Port: 6001, Dst Port: 6001										
Datagram Transport Layer Security										
- DTLSv1.2 Record Layer: Handshake Protocol: Server Hello										
Content Type: Handshake (22)										
Version: DTLS 1.2 (0xffffd)										
Epoch: 0										
Sequence Number: 1										
Length: 73										
- Handshake Protocol: Server Hello										
Handshake Type: Server Hello (2)										
Length: 61										
Message Sequence: 1										
Fragment Offset: 0										
Fragment Length: 61										
Version: DTLS 1.2 (0xffffd)										
- Random: 23746d4d8d4136c1e34c965753d673ea6761772c7ef58bd68257f075c59260fe										
Session ID Length: 0										
- Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0x0c2c)										
Compression Method: null (0)										
Extensions Length: 21										
- Extension: renegotiation_info (len=1)										
- Extension: ec_point_formats (len=4)										
- Extension: session_ticket (len=0)										
- Extension: extended_master_secret (len=0)										
[JAS3 Fullstring: 65277, 491996, 65281-11-35-23]										
[JAS5: \$te93b4dc7715b5562a0e94a9fc]										
- DTLSv1.2 Record Layer: Handshake Protocol: Certificate										
- DTLSv1.2 Record Layer: Handshake Protocol: Server Key Exchange										
- DTLSv1.2 Record Layer: Handshake Protocol: Certificate Request										
- DTLSv1.2 Record Layer: Handshake Protocol: Server Hello Done										
0067	19:14:58.202000000	127.0.1.1	6001	127.0.1.1	5001	DTLSv1.2	00	00 15 ff 01 00 01 00	u`...`.	.....
0068	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	23 00 00 17 00 00 00	u`...`.	.....
0069	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0070	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0071	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0072	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0073	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0074	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0075	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0076	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0077	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0078	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0079	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0080	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0081	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0082	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0083	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0084	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0085	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0086	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0087	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0088	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0089	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0090	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0091	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0092	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0093	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0094	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0095	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0096	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0097	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0098	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0099	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0100	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0101	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0102	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0103	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0104	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0105	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0106	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0107	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0108	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0109	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0110	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0111	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0112	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0113	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0114	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0115	19:14:58.202000000	127.0.1.1	5001	127.0.1.1	6001	DTLSv1.2	00	00 00 00 00 00 00 00	u`...`.	.....
0116	19:14:58.202000000									

**Part of the code used for conveying the selected cipher suite to Client through ‘Server Hello’:**

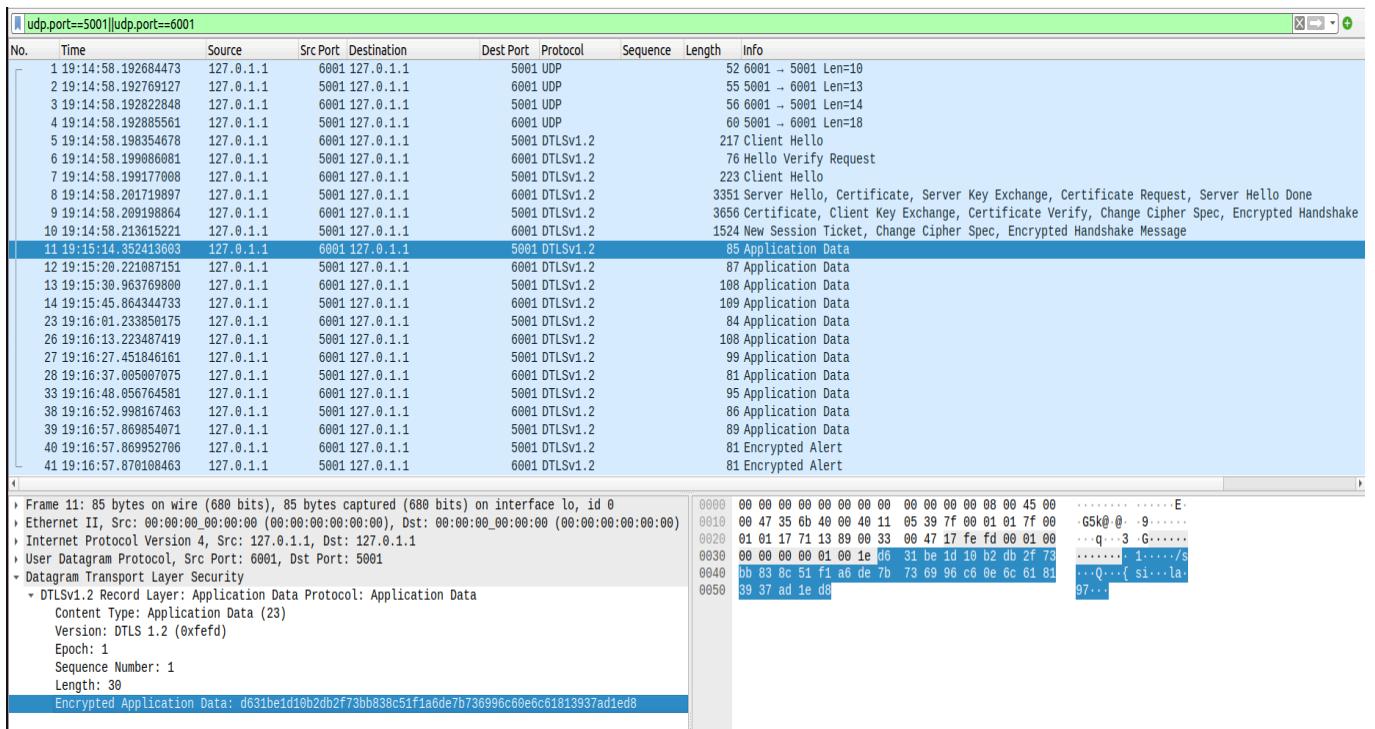
```
SSL_CTX *create_server_context() // bob
{
    // Creating context and filling certs and private key in it
    SSL_CTX *ctx = SSL_CTX_new(DTLSv1_2_server_method());

    // Setting the cipher suits list
    SSL_CTX_set_cipher_list(ctx, "ECDHE-ECDSA-AES256-GCM-SHA384");
```

This is Server side code, which sets the selected cipher suites in the DTLS 1.2 context of server.

#### d) Encrypted chatting using DTLS 1.2:

- We can observe in the below screenshot, chat messages between Client and Server are exchanged in the encrypted form.
- As can be seen from the protocol column in wireshark, the client and server uses DTLS version 1.2 protocol for chatting
- The ‘Application data’ packets, contains the encrypted chat messages.



**e) Difference between DTLS 1.2 handshake and TLS 1.2 handshake:**

Comparison	DTLS 1.2	TLS 1.2
1) Client Hello	<p>In DTLS 1.2 two “ClientHello” messages are exchanged:</p> <ul style="list-style-type: none"> <li>For the first “ClientHello” message, the server’s response includes a “HelloVerifyRequest” message and a cookie.</li> <li>Client then sends the second “ClientHello” message along with the received cookie, which then the server verifies and responds with the “ServerHello” message</li> </ul>	In this case, there is only one “ClientHello” message sent by client to the server, which in turn responds with the “ServerHello” message (containing the Server Certificate, ServerKeyExchange, CertificateRequest).
2) Connection Type	It uses UDP as the underlying transmission protocol.	It uses TCP as the underlying transmission protocol.
3) Protection Against IP Spoofing	Uses cookies to prevent IP spoofing	Performs communications only after establishing the TCP handshake. This makes it difficult to spoof.
4) Reliability and ordered delivery	Does not guarantee reliable or ordered delivery of packets	Provides reliable, ordered, and error-checked delivery of packets
5) Fragmentation and Retransmission	It handles fragmentation and retransmission at the application layer because UDP lacks built-in support for these features. DTLS include mechanisms such as message retransmission, ack and timeout handling to ensure handshake reliability.	It relies on TCP for fragmentation and retransmission of lost packets, or packets arriving out of order.
6) Latency	Reduces latency and overhead. Hence it is more suitable for real time applications such as VoIP, etc.	Used when data integrity and ordering is more important than reducing latency.

## g) Chat closing scenarios:

### Scenario 1: When client closes the chat

- Client can close the chatting by sending the 'chat\_close' message to Server.
- When the Client closes the chat, the Server keeps on listening for incoming connection requests from newer Clients.

```
-----  
Server connected to Client using DTLS 1.2  
-----  
***** Server has started chatting with Client *****  
  
From Client: Hi Bobl  
Enter message for Client: Hi Alice1  
  
From Client: The weather seems good today!  
Enter message for Client: Yes. Can we go out for picnic?  
  
From Client: Sure.  
Enter message for Client: Tomorrow 8 am is fine for you?  
  
From Client: Yes, lets go by car.  
Enter message for Client: ok  
  
From Client: See you tomorrow  
Enter message for Client: bye  
  
From Client: chat_close  
***** Chat closed *****  
  
-----  
Server socket created successfully  
-----  
Server started listening for connection....  
  
-----  
Client connected to Server using DTLS  
-----  
***** Client has started chatting with Server *****  
  
Enter message for Server: Hi Bobl  
  
From Server: Hi Alice1  
Enter message for Server: The weather seems good today!  
  
From Server: Yes. Can we go out for picnic?  
Enter message for Server: Sure.  
  
From Server: Tomorrow 8 am is fine for you?  
Enter message for Server: Yes, lets go by car.  
  
From Server: ok  
Enter message for Server: See you tomorrow  
  
From Server: bye  
Enter message for Server: chat_close  
  
***** Chat closed *****  
  
-----  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$
```

### Scenario 2: When server closes the chat

- Server can also close the chatting by sending the 'chat\_close' message to Client.
- When the Server closes the chat with one client, it keeps on listening for incoming connection requests from newer Clients.

```
-----  
Server connected to Client using DTLS 1.2  
-----  
***** Server has started chatting with Client *****  
  
From Client: Hi Bobl  
Enter message for Client: Hi Alice1  
  
From Client: How are you?  
Enter message for Client: I am fine  
  
From Client: nice  
Enter message for Client: chat_close  
  
***** Chat closed *****  
  
-----  
Server socket created successfully  
-----  
Server started listening for connection....  
[]  
  
-----  
Client connected to Server using DTLS  
-----  
***** Client has started chatting with Server *****  
  
Enter message for Server: Hi Bobl  
  
From Server: Hi Alice1  
Enter message for Server: How are you?  
  
From Server: I am fine  
Enter message for Server: nice  
  
From Server: chat_close  
***** Chat closed *****  
  
-----  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$  
mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$
```

## Packet Loss:

1. Command to see all connected interfaces, used below command:

```
$ ifconfig
```

```
root@alice1:~# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 172.31.0.2 netmask 255.255.255.0 broadcast 172.31.0.255
        inet6 fe80::216:3eff:feae:c3fd prefixlen 64 scopeid 0x20<link>
          ether 00:16:3e:ae:c3:fd txqueuelen 1000 (Ethernet)
            RX packets 114587 bytes 537006277 (537.0 MB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 82479 bytes 7202690 (7.2 MB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
            RX packets 1599 bytes 142528 (142.5 KB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 1599 bytes 142528 (142.5 KB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

```
root@alice1:~#
```

2. Command used to View the Configured Network Conditions:

```
$ sudo tc qdisc show dev <interface_name>
```

```
root@alice1:~# sudo tc qdisc show dev eth0
qdisc noqueue 0: root refcnt 2
```

3. Command used to inject the packet loss:

```
$ sudo tc qdisc add dev <interface_name> root netem loss 33%
```

```
root@alice1:~# sudo tc qdisc add dev eth0 root netem loss 55%
root@alice1:~# sudo tc qdisc show dev eth0
qdisc netem 8026: root refcnt 3 limit 1000 loss 55%
root@alice1:~#
```

```
root@bob1:~# sudo tc qdisc add dev eth0 root netem loss 55%
root@bob1:~# sudo tc qdisc show dev eth0
qdisc netem 8027: root refcnt 3 limit 1000 loss 55%
root@bob1:~#
```

#### 4. Command used to delete the packet loss configuration:

```
$ sudo tc qdisc del dev <interface_name> root netem
```

```
root@bob1:~# sudo tc qdisc del dev eth0 root netem
root@bob1:~#
```

#### Secure chat after injecting packet loss at alice1 and bob1:

- We can observe that, after injecting 55% packet loss in the application layer, some plaintext control messages get lost, and are **retransmitted to achieve reliability**.
- Retransmitted control messages are highlighted using red color.

```
File Edit Selection View Go Run Terminal Help
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
home [SSH: 10.200.32.196]
root@alice1:~# ./secure_chat_app -c bob1
Client will chat with server:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001
-----
Client Info:
Client IP: 172.31.0.2
Client Port: 6001
-----
Client socket created Successfully
-----
Msg sent to Server: chat_hello
-----
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
[Packet lost. Resent 'chat_START_SSL' message to server....]
[Packet lost. Resent 'chat_START_SSL' message to server....]
[Packet lost. Resent 'chat_START_SSL' message to server....]
-----
Msg recv from Server: chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
Depth:0, Cert Info: /CN=Bob1.com/emailAddress=Bob1@gmail.com/O=Bob1/L=Kolkata/C=IN
Certificate having depth 0 is NOT expired
-----
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
-----
.... DTLS 1.2 Handshake ended ....
Client connected to Server using DTLS
-----
***** Client has started chatting with Server *****
Enter message for Server: HI Bob1
From Server: Hi alice1
Enter message for Server: How are you?
From Server: I am fine.
Enter message for Server: How is weather in london?
From Server: chat_close
***** Chat closed *****

root@bob1:~# ./secure_chat_app -s
Server Info:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for connection....
-----
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
[Packet lost. Resent 'chat_ok_reply' message to client....]
-----
Msg recv from Client: chat_START_SSL
Msg sent to Client: chat_START_SSL_ACK
[Packet lost. Resent chat_START_SSL_ACK message to client....]
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
Depth:0, Cert Info: /CN=Alicel.com/emailAddress=Alice1@gmail.com/O=Alice1/L=Nagpur/C=IN
Certificate having depth 0 is NOT expired
-----
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
-----
.... DTLS 1.2 Handshake ended ....
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****
From Client: HI Bob1
Enter message for Client: Hi alice1
From Client: How are you?
Enter message for Client: I am fine.
From Client: How is weather in london?
Enter message for Client: chat_close
*****
----- Chat closed -----
-----
Server socket created successfully
```

## Wireshark trace after injecting packet loss:

### Tcpdump trace at alice1:

alice1\_reliability\_task2.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-/>

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.3	5001	UDP	52	6001 → 5001 Len=10
2	0.000018	172.31.0.3	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
3	1.001625	172.31.0.3	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
4	1.001818	172.31.0.2	6001	172.31.0.3	5001	UDP	56	6001 → 5001 Len=14
5	2.025522	172.31.0.2	6001	172.31.0.3	5001	UDP	56	6001 → 5001 Len=14
6	2.025754	172.31.0.3	5001	172.31.0.2	6001	UDP	60	5001 → 6001 Len=18
7	2.029470	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	217	Client Hello
8	5.065467	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	217	Client Hello
9	5.065680	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	76	Hello Verify Request
10	5.065941	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	223	Client Hello
11	5.071849	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	1514	Server Hello, Certificate (Fragment)
12	5.071910	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	1514	Certificate (Fragment)
13	5.072512	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	456	Certificate (Reassembled), Server Key Exchange, Certificate Request, Server Hello Done
14	5.085780	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	1514	Certificate (Fragment)
15	5.085815	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	1514	Certificate (Fragment)
16	5.087467	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	762	Certificate (Reassembled), Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
17	5.091376	172.31.0.3	172.31.0.2	5001	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=bde1) [Reassembled in #18]	
18	5.091414	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	44	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
19	5.129310	Xensource_ae:c3:fd	Xensource_d2:a2:f0	ARP	42	Who has 172.31.0.3? Tell 172.31.0.2		
20	5.129512	Xensource_ae:c3:fd	Xensource_d2:a2:f0	ARP	42	Who has 172.31.0.2? Tell 172.31.0.3		
21	5.129523	Xensource_ae:c3:fd	Xensource_d2:a2:f0	ARP	42	172.31.0.2 is at 00:16:3e:ae:c3:fd		
22	5.129530	Xensource_d2:a2:f0	Xensource_ae:c3:fd	ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0		

> Frame 6: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)  
> Ethernet II, Src: Xensource\_d2:a2:f0 (00:16:3e:d2:a2:f0), Dst: Xensource\_ae:c3:fd (00:16:3e:ae:c3:fd)  
> Internet Protocol Version 4, Src: 172.31.0.3, Dst: 172.31.0.2  
> User Datagram Protocol, Src Port: 6001, Dst Port: 6001  
> Data (18 bytes)

0000 00 16 3e ae c3 fd 00 16 3e d2 a2 f0 00 00 45 00 ...>.....>....E.  
0010 00 2e 0f 8d 40 00 40 11 d2 ee ac 1f 00 03 ac 1f ..@ @ ..  
0020 00 02 13 89 17 71 00 1a 58 6f 63 68 61 74 5f 53 ...q... Xchat\_S  
0030 54 41 52 54 5f 53 53 4c 5f 41 43 4b TART\_SSL\_ACK

### Tcpdump trace at bob1:

bob1\_reliability\_task2.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter... <Ctrl-/>

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.3	5001	UDP	52	6001 → 5001 Len=10
2	0.000093	172.31.0.3	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
3	1.001580	172.31.0.3	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
4	1.001821	172.31.0.2	6001	172.31.0.3	5001	UDP	56	6001 → 5001 Len=14
5	2.025528	172.31.0.2	6001	172.31.0.3	5001	UDP	56	6001 → 5001 Len=14
6	2.025719	172.31.0.3	5001	172.31.0.2	6001	UDP	60	5001 → 6001 Len=18
7	2.029474	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	217	Client Hello
8	5.065471	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	217	Client Hello
9	5.065641	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	76	Hello Verify Request
10	5.065931	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	223	Client Hello
11	5.071816	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	1514	Server Hello, Certificate (Fragment)
12	5.071886	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	1514	Certificate (Fragment)
13	5.072487	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	456	Certificate (Reassembled), Server Key Exchange, Certificate Request, Server Hello Done
14	5.085768	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	1514	Certificate (Fragment)
15	5.085798	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	1514	Certificate (Fragment)
16	5.087452	172.31.0.2	6001	172.31.0.3	5001	DTLSv1.2	762	Certificate (Reassembled), Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
17	5.091347	172.31.0.3	172.31.0.2	5001	IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=bde1) [Reassembled in #18]	
18	5.091393	172.31.0.3	5001	172.31.0.2	6001	DTLSv1.2	44	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
19	5.129257	Xensource_d2:a2:f0	Xensource_ae:c3:fd	ARP	42	Who has 172.31.0.2? Tell 172.31.0.3		
20	5.129495	Xensource_ae:c3:fd	Xensource_d2:a2:f0	ARP	42	Who has 172.31.0.3? Tell 172.31.0.2		
21	5.129509	Xensource_d2:a2:f0	Xensource_ae:c3:fd	ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0		
22	5.129511	Xensource_ae:c3:fd	Xensource_d2:a2:f0	ARP	42	172.31.0.2 is at 00:16:3e:ae:c3:fd		

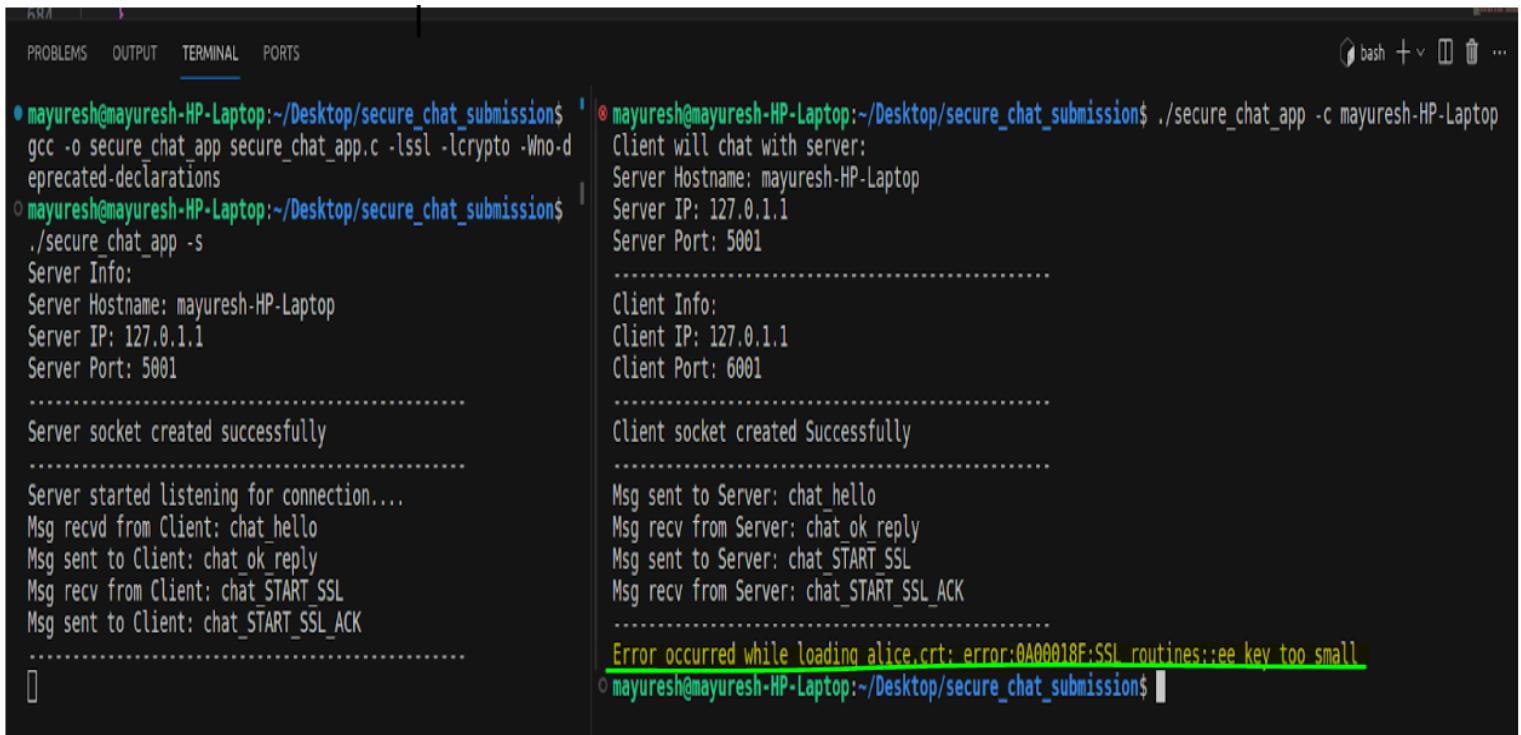
> Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits)  
> Ethernet II, Src: Xensource\_ae:c3:fd (00:16:3e:ae:c3:fd), Dst: Xensource\_d2:a2:f0 (00:16:3e:d2:a2:f0)  
> Internet Protocol Version 4, Src: 172.31.0.2, Dst: 172.31.0.3  
> User Datagram Protocol, Src Port: 6001, Dst Port: 5001  
> Data (18 bytes)

0000 00 16 3e d2 a2 f0 00 16 3e ae c3 fd 00 00 45 00 ...>.....>....E.  
0010 00 26 28 25 40 00 40 11 ba 5e ac 1f 00 02 ac 1f %@ ..  
0020 00 03 17 71 13 89 00 12 58 6f 63 68 61 74 5f 68 ...q... Xchat\_S  
0030 65 6c 6c 6f elio

From the above pcap trace at alice1 and bob1, we can see that plain text control messages are being retransmitted due to packet loss.

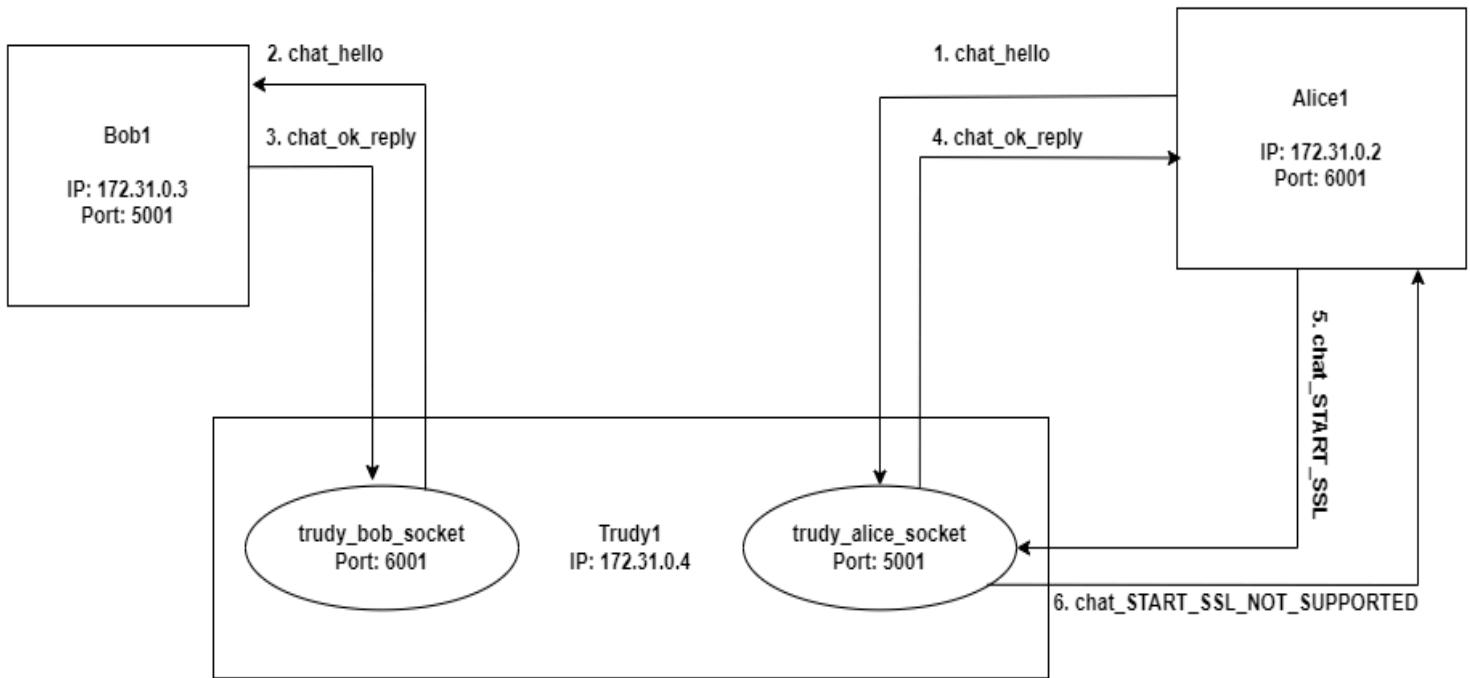
### **Challenges faced during the implementation of this secure\_chat\_app:**

The private key generated for Alice was a 1024-bit RSA key. However, when we were trying to load this private key during the code execution we encountered an error “ee key too small”. Therefore for finally running the code we have used a 2048-bit RSA key. This has been kept in the “alice\_2048” directory.



```
PROBLEMS OUTPUT TERMINAL PORTS bash + v ... ● mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ gcc -o secure_chat_app secure_chat_app.c -lssl -lcrypto -Wno-d deprecated-declarations ○ mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -s
Server Info:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for connection....
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok reply
Msg recv from Client: chat_START_SSL
Msg sent to Client: chat_START_SSL_ACK
-----
○ mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ ./secure_chat_app -c mayuresh-HP-Laptop
Client will chat with server:
Server Hostname: mayuresh-HP-Laptop
Server IP: 127.0.1.1
Server Port: 5001
-----
Client Info:
Client IP: 127.0.1.1
Client Port: 6001
-----
Client socket created Successfully
-----
Msg sent to Server: chat_hello
Msg recv from Server: chat_ok reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_ACK
-----
○ mayuresh@mayuresh-HP-Laptop:~/Desktop/secure_chat_submission$ Error occurred while loading alice.crt: error:0A00018F:SSL routines::ee key too small
```

### **TASK 3: START SSL downgrade attack for eavesdropping**



**Fig: Control Message flow diagram of downgrade attack**

- When the downgrade attack is performed by trudy1, the entire chatting happens in plain text messages through **UDP sockets (SOCK\_DGRAM)**.
- Here, trudy1 acts as a passive attacker, i.e., trudy1 only eavesdrops on the chat messages sent between alice1 and bob1, but does not modify their content.
- To act as a passive attacker, it poisons the /etc/hosts file by replacing the IP addresses of alice1 with bob1 and vice versa.
- When alice1 sends a 'chat\_START\_SSL' message to bob1, it gets intercepted by trudy1.
- Now trudy1, sends 'chat\_START\_SSL\_NOT\_SUPPORTED' message to alice1, hence performing the 'Downgrade attack'.
- From now the entire communication between alice1 and bob1 happens in plain text. Hence, trudy1 can easily eavesdrop on chat messages.

## Downgrade attack on secure chat output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

root@alice1:~# ./secure_chat_app -c bob1
Current Client Info:
Client IP: 172.31.0.2
Client Port: 6001

Client socket created Successfully

Client will chat with server:
Server Hostname: bob1
Server IP (after DNS poisoning): 172.31.0.4
Server Port: 5001

Msg sent to Server: chat_hello
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_NOT_SUPPORTED

***** (Unsecured) Chat started *****
Enter message for Server: Hi bob1, how are you?
Msg sent to Server: Hi bob1, how are you?
Msg recv from Server: I am fine alice1.

Enter message for Server: Can we go for dinner tonight?
Msg sent to Server: Can we go for dinner tonight?
Msg recv from Server: Yes. Which place?

Enter message for Server: Santosh hotel?
Msg sent to Server: Santosh hotel?
Msg recv from Server: Ok. See you there!

Enter message for Server: chat_close
Msg sent to Server: chat_close

***** Chat closed *****
root@alice1:~#

```

```

root@trudy1:~# ./secure_chat_interceptor -d alice1 bob1
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4

'trudy_alice_socket' socket created Successfully

Waiting for 'chat_hello' msg from Alice1...
Msg recv from alice1 (172.31.0.2:28951): chat_hello

'trudy_bob_socket' socket created Successfully

Msg sent to bob1 (172.31.0.3:5001): chat_hello
Msg recv from bob1 (172.31.0.3:5001): chat_ok_reply
Msg sent to alice1 (172.31.0.2:28951): chat_OK_REPLY
Msg recv from alice1 (172.31.0.2:28951): chat_START_SSL
Msg sent to bob1 (172.31.0.3:5001): chat_START_SSL_NOT_SUPPORTED
Msg recv from bob1 (172.31.0.3:5001): UNENCRYPTED_CHAT_ACK
Msg sent to alice1 (172.31.0.2:28951): chat_START_SSL_NOT_SUPPORTED

***** Eavesdropping started *****
Msg recv from alice1 (172.31.0.2:28951): Hi bob1, how are you?
Msg sent to bob1 (172.31.0.3:5001): Hi bob1, how are you?

Msg recv from bob1 (172.31.0.3:5001): I am fine alice1.
Msg sent to alice1 (172.31.0.2:28951): I am fine alice1.

Msg recv from alice1 (172.31.0.2:28951): Can we go for dinner tonight?
Msg sent to bob1 (172.31.0.3:5001): Can we go for dinner tonight?

Msg recv from bob1 (172.31.0.3:5001): Yes. Which place?
Msg sent to alice1 (172.31.0.2:28951): Yes. Which place?

Msg recv from alice1 (172.31.0.2:28951): Santosh hotel?
Msg sent to bob1 (172.31.0.3:5001): Santosh hotel?

Msg recv from bob1 (172.31.0.3:5001): Ok. See you there!
Msg sent to alice1 (172.31.0.2:28951): Ok. See you there!

Msg recv from alice1 (172.31.0.2:28951): chat_close
Msg sent to bob1 (172.31.0.3:5001): chat_close

***** Chat closed by 'alice1' *****

Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4

'trudy_alice_socket' socket created Successfully

Waiting for 'chat_hello' msg from Alice1...
[]


```

```

root@bob1:~# ./secure_chat_app -s
Server Info:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001

Server socket created successfully
-----
Server started listening for new client connection...
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
Msg recv from Client: chat_START_SSL_NOT_SUPPORTED
Msg sent to Client: UNENCRYPTED_CHAT_ACK

***** (Unsecured) Chat started *****
Waiting for chat msg from Client...
Msg recv from Client: Hi bob1, how are you?

Enter message for Client: I am fine alice1.
Msg sent to Client: I am fine alice1.
Msg recv from Client: Can we go for dinner tonight?

Enter message for Client: Yes. Which place?
Msg sent to Client: Yes. Which place?
Msg recv from Client: Santosh hotel?

Enter message for Client: Ok. See you there!
Msg sent to Client: Ok. See you there!
Msg recv from Client: chat_close

***** Chat closed *****
Server socket created successfully
-----
Server started listening for new client connection...
[]


```

- When the chat is closed by anyone (client or server), then the server starts listening for new client connection requests.
- When the chat is closed, trudy1 keeps on running to eavesdrop any upcoming client server communication that will occur in future.

**Downgrade attack on secure chat ‘alice1’ output:**

```
root@alice1:~# ./secure_chat_app -c bob1
Current Client Info:
Client IP: 172.31.0.2
Client Port: 6001

-----
Client socket created Successfully

-----
Client will chat with server:
Server Hostname: bob1
Server IP (after DNS poisoning): 172.31.0.4
Server Port: 5001

-----
Msg sent to Server: chat_hello
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_NOT_SUPPORTED

***** (Unsecured) Chat started *****
Enter message for Server: Hi bob1, how are you?
Msg sent to Server: Hi bob1, how are you?
Msg recv from Server: I am fine alice1.

Enter message for Server: Can we go for dinner tonight?
Msg sent to Server: Can we go for dinner tonight?
Msg recv from Server: Yes. Which place?

Enter message for Server: Santosh hotel?
Msg sent to Server: Santosh hotel?
Msg recv from Server: Ok. See you there!

Enter message for Server: chat_close
Msg sent to Server: chat_close

***** Chat closed *****
root@alice1:~# █
```

**Downgrade attack on secure chat ‘bob1’ output:**

```
root@bob1:~# ./secure_chat_app -s
Server Info:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for new client connection....
Msg recv from Client: chat_hello
Msg sent to Client: chat_ok_reply
Msg recv from Client: chat_START_SSL_NOT_SUPPORTED
Msg sent to Client: UNENCRYPTED_CHAT_ACK

***** (Unsecured) Chat started *****
Waiting for chat msg from Client....
Msg recv from Client: Hi bob1, how are you?

Enter message for Client: I am fine alice1.
Msg sent to Client: I am fine alice1.
Msg recv from Client: Can we go for dinner tonight?

Enter message for Client: Yes. Which place?
Msg sent to Client: Yes. Which place?
Msg recv from Client: Santosh hotel?

Enter message for Client: Ok. See you there!
Msg sent to Client: Ok. See you there!
Msg recv from Client: chat_close

***** Chat closed *****
Server socket created successfully
-----
Server started listening for new client connection....
```

## Downgrade attack on secure chat ‘trudy1’ output:

```
root@trudy1:~# ./secure_chat_interceptor -d alice1 bob1
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4
-----
'trudy_alice_socket' socket created Successfully
-----
Waiting for 'chat_hello' msg from Alice1.....
Msg recv from alice1 (172.31.0.2:28951): chat_hello
-----
'trudy_bob_socket' socket created Successfully
-----
Msg sent to bob1 (172.31.0.3:5001): chat_hello
Msg recv from bob1 (172.31.0.3:5001): chat_ok_reply
Msg sent to alice1 (172.31.0.2:28951): chat_ok_reply
Msg recv from alice1 (172.31.0.2:28951): chat_START_SSL
Msg sent to bob1 (172.31.0.3:5001): chat_START_SSL_NOT_SUPPORTED
Msg recv from bob1 (172.31.0.3:5001): UNENCRYPTED_CHAT_ACK
Msg sent to alice1 (172.31.0.2:28951): chat_START_SSL_NOT_SUPPORTED

***** Evesdropping started *****
Msg recv from alice1 (172.31.0.2:28951): Hi bob1, how are you?
Msg sent to bob1 (172.31.0.3:5001): Hi bob1, how are you?

Msg recv from bob1 (172.31.0.3:5001): I am fine alice1.
Msg sent to alice1 (172.31.0.2:28951): I am fine alice1.

Msg recv from alice1 (172.31.0.2:28951): Can we go for dinner tonight?
Msg sent to bob1 (172.31.0.3:5001): Can we go for dinner tonight?

Msg recv from bob1 (172.31.0.3:5001): Yes. Which place?
Msg sent to alice1 (172.31.0.2:28951): Yes. Which place?

Msg recv from alice1 (172.31.0.2:28951): Santosh hotel?
Msg sent to bob1 (172.31.0.3:5001): Santosh hotel?

Msg recv from bob1 (172.31.0.3:5001): Ok. See you there!
Msg sent to alice1 (172.31.0.2:28951): Ok. See you there!

Msg recv from alice1 (172.31.0.2:28951): chat_close
Msg sent to bob1 (172.31.0.3:5001): chat_close

***** Chat closed by 'alice1' *****

-----
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4
-----
'trudy_alice_socket' socket created Successfully
-----
Waiting for 'chat_hello' msg from Alice1.....
[]
```

## 1. Tcpdump trace captured at 'alice1':

alice1\_task3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 172.31.0.2

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10
3	0.000715	172.31.0.4	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
4	0.001022	172.31.0.2	6001	172.31.0.4	5001	UDP	56	6001 → 5001 Len=14
5	0.001269	172.31.0.4	5001	172.31.0.2	6001	UDP	70	5001 → 6001 Len=28
10	15.144380	172.31.0.2	6001	172.31.0.4	5001	UDP	63	6001 → 5001 Len=21
11	32.025900	172.31.0.4	5001	172.31.0.2	6001	UDP	59	5001 → 6001 Len=17
12	49.316410	172.31.0.2	6001	172.31.0.4	5001	UDP	71	6001 → 5001 Len=29
15	63.880459	172.31.0.4	5001	172.31.0.2	6001	UDP	59	5001 → 6001 Len=17
18	77.829738	172.31.0.2	6001	172.31.0.4	5001	UDP	56	6001 → 5001 Len=14
19	95.042559	172.31.0.4	5001	172.31.0.2	6001	UDP	60	5001 → 6001 Len=18
20	107.388213	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10

```
> Frame 1: 52 bytes on wire (416 bits), 52 bytes captured (416 bits)
> Ethernet II, Src: Xensource_ae:c3:fd (00:16:3e:ae:c3:fd), Dst: Xensource_3d:17:94 (00:16:3e:3d:17:94)
> Internet Protocol Version 4, Src: 172.31.0.2, Dst: 172.31.0.4
> User Datagram Protocol, Src Port: 6001, Dst Port: 5001
    Source Port: 6001
    Destination Port: 5001
    Length: 18
    Checksum: 0x5868 [unverified]
        [Checksum Status: Unverified]
        [Stream index: 0]
    > [Timestamps]
        UDP payload (10 bytes)
    < Data (10 bytes)
        Data: 636861745f68656c6c0f
        [Length: 10]
```

0000 00 16 3e 3d 17 94 00 16 3e ae c3 fd 08 00 45 00 ...>....>....E-
0010 00 26 17 6a 40 00 40 11 cb 18 ac 1f 00 02 ac 1f ..& j@. @. ....
0020 00 04 17 71 13 89 00 12 58 68 63 68 61 74 5f 68 ...q....Xchat\_h
0030 65 6c 6c 6f hello

alice1\_task3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 172.31.0.2

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10
3	0.000715	172.31.0.4	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
4	0.001022	172.31.0.2	6001	172.31.0.4	5001	UDP	56	6001 → 5001 Len=14
5	0.001269	172.31.0.4	5001	172.31.0.2	6001	UDP	70	5001 → 6001 Len=28
10	15.144380	172.31.0.2	6001	172.31.0.4	5001	UDP	63	6001 → 5001 Len=21
11	32.025900	172.31.0.4	5001	172.31.0.2	6001	UDP	59	5001 → 6001 Len=17
12	49.316410	172.31.0.2	6001	172.31.0.4	5001	UDP	71	6001 → 5001 Len=29
15	63.880459	172.31.0.4	5001	172.31.0.2	6001	UDP	59	5001 → 6001 Len=17
18	77.829738	172.31.0.2	6001	172.31.0.4	5001	UDP	56	6001 → 5001 Len=14
19	95.042559	172.31.0.4	5001	172.31.0.2	6001	UDP	60	5001 → 6001 Len=18
20	107.388213	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10

```
> Frame 5: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Xensource_3d:17:94 (00:16:3e:3d:17:94), Dst: Xensource_ae:c3:fd (00:16:3e:ae:c3:fd)
> Internet Protocol Version 4, Src: 172.31.0.4, Dst: 172.31.0.2
> User Datagram Protocol, Src Port: 5001, Dst Port: 6001
    Source Port: 5001
    Destination Port: 6001
    Length: 36
    Checksum: 0x587a [unverified]
        [Checksum Status: Unverified]
        [Stream index: 0]
    > [Timestamps]
        UDP payload (28 bytes)
    < Data (28 bytes)
        Data: 636861745f53544152545f53534c5f4e4f545f535550504f52544544
        [Length: 28]
```

0000 00 16 3e ae c3 fd 00 16 3e 3d 17 94 08 00 45 00 ...>....>....E-
0010 00 38 fd 26 40 00 40 11 e5 49 ac 1f 00 04 ac 1f 8 @@. @. I....
0020 00 02 13 89 17 71 00 24 58 7a 63 68 61 74 5f 53 .....q \$ Xchat\_S
0030 54 41 52 54 5f 53 53 4c 5f 4e 4f 54 5f 53 55 50 TART\_SSL\_NOT\_SUPPORTED
0040 50 4f 52 54 45 44

In this pcap, we can observe the 'chat\_START\_SSL\_NOT\_SUPPORTED' msg sent by trudy1 to alice1.

## 2. Tcpdump trace captured at ‘bob1’:

bob1\_task3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 172.31.0.3

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
2	0.000307	172.31.0.4		28951 172.31.0.3		5001 UDP	52	28951 → 5001 Len=10
3	0.000504	172.31.0.3		5001 172.31.0.4		28951 UDP	55	5001 → 28951 Len=13
4	0.000995	172.31.0.4		28951 172.31.0.3		5001 UDP	70	28951 → 5001 Len=28
5	0.001063	172.31.0.3		5001 172.31.0.4		28951 UDP	62	5001 → 28951 Len=20
10	15.144550	172.31.0.4		28951 172.31.0.3		5001 UDP	63	28951 → 5001 Len=21
11	32.025482	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
12	49.316580	172.31.0.4		28951 172.31.0.3		5001 UDP	71	28951 → 5001 Len=29
15	63.879899	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
18	77.830550	172.31.0.4		28951 172.31.0.3		5001 UDP	56	28951 → 5001 Len=14
19	95.042097	172.31.0.3		5001 172.31.0.4		28951 UDP	60	5001 → 28951 Len=18
20	107.389449	172.31.0.4		28951 172.31.0.3		5001 UDP	52	28951 → 5001 Len=10

```
> Frame 19: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: Xensource_d2:a2:f0 (00:16:3e:d2:a2:f0), Dst: Xensource_3d:17:94 (00:16:3e:3d:17:94)
> Internet Protocol Version 4, Src: 172.31.0.3, Dst: 172.31.0.4
> User Datagram Protocol, Src Port: 5001, Dst Port: 28951
> Data (18 bytes)
```

0000	00 16 3e 3d 17 94 00 16 3e d2 a2 f0 08 00 45 00	...=.... >....E
0010	00 2e b7 14 40 00 40 11 2b 65 ac 1f 00 03 ac 1f	...@...+e.....
0020	00 04 13 89 71 17 00 1a 58 71 4f 6b 2e 20 53 65	....q... Xo0k. Se
0030	65 20 79 6f 75 20 74 68 65 72 65 21	e you th ere!

- We can observe that the chat messages are transferred in the plain text from this pcap trace.
- After receiving the ‘chat\_close’ message from the client, the bob1 (server) closed the current client connection and started listening for new client connection requests.

bob1\_task3.pcap

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

ip.addr == 172.31.0.3

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
2	0.000307	172.31.0.4		28951 172.31.0.3		5001 UDP	52	28951 → 5001 Len=10
3	0.000504	172.31.0.3		5001 172.31.0.4		28951 UDP	55	5001 → 28951 Len=13
4	0.000995	172.31.0.4		28951 172.31.0.3		5001 UDP	70	28951 → 5001 Len=28
5	0.001063	172.31.0.3		5001 172.31.0.4		28951 UDP	62	5001 → 28951 Len=20
10	15.144550	172.31.0.4		28951 172.31.0.3		5001 UDP	63	28951 → 5001 Len=21
11	32.025482	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
12	49.316580	172.31.0.4		28951 172.31.0.3		5001 UDP	71	28951 → 5001 Len=29
15	63.879899	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
18	77.830550	172.31.0.4		28951 172.31.0.3		5001 UDP	56	28951 → 5001 Len=14
19	95.042097	172.31.0.3		5001 172.31.0.4		28951 UDP	60	5001 → 28951 Len=18
20	107.389449	172.31.0.4		28951 172.31.0.3		5001 UDP	52	28951 → 5001 Len=10

```
> Frame 20: 52 bytes on wire (416 bits), 52 bytes captured (416 bits)
> Ethernet II, Src: Xensource_3d:17:94 (00:16:3e:3d:17:94), Dst: Xensource_d2:a2:f0 (00:16:3e:d2:a2:f0)
> Internet Protocol Version 4, Src: 172.31.0.4, Dst: 172.31.0.3
> User Datagram Protocol, Src Port: 28951, Dst Port: 5001
> Data (10 bytes)
```

0000	00 16 3e d2 a2 f0 00 16 3e 3d 17 94 08 00 45 00	...>.... >....E
0010	00 26 9c 6f 40 00 40 11 46 12 ac 1f 00 04 ac 1f	...&@...F.....
0020	00 03 71 17 13 89 00 12 58 69 63 68 61 74 5f 63	....q... Xchat_c
0030	6c 6f 73 65	lose

### 3. Tcpdump trace captured at 'trudy1':

trudy1\_task3.pcap

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2		6001 172.31.0.4		5001 UDP	52	6001 → 5001 Len=10
2	0.000246	172.31.0.4		28951 172.31.0.3		5001 UDP	52	28951 → 5001 Len=10
3	0.000526	172.31.0.3		5001 172.31.0.4		28951 UDP	55	5001 → 28951 Len=13
4	0.000591	172.31.0.4		5001 172.31.0.2		6001 UDP	55	5001 → 6001 Len=13
5	0.000911	172.31.0.2		6001 172.31.0.4		5001 UDP	56	6001 → 5001 Len=14
6	0.000990	172.31.0.4		28951 172.31.0.3		5001 UDP	70	28951 → 5001 Len=28
7	0.001071	172.31.0.3		5001 172.31.0.4		28951 UDP	62	5001 → 28951 Len=20
8	0.001145	172.31.0.4		5001 172.31.0.2		6001 UDP	70	5001 → 6001 Len=28
9	5.217855	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	Who has 172.31.0.2? Tell 172.31.0.4
10	5.217927	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.2
11	5.217934	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3? Tell 172.31.0.4
12	5.217948	Xensource_d2:a2:f0		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
13	5.218389	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.3
14	5.218422	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
15	5.218441	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	172.31.0.2 is at 00:16:3e:a:c3:fd
16	5.218446	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0
17	15.144285	172.31.0.2		6001 172.31.0.4		5001 UDP	63	6001 → 5001 Len=21
18	15.144529	172.31.0.4		28951 172.31.0.3		5001 UDP	63	28951 → 5001 Len=21
19	32.025508	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
20	32.025769	172.31.0.4		5001 172.31.0.2		6001 UDP	59	5001 → 6001 Len=17
21	49.316315	172.31.0.2		6001 172.31.0.4		5001 UDP	71	6001 → 5001 Len=29
22	49.316553	172.31.0.4		28951 172.31.0.3		5001 UDP	71	28951 → 5001 Len=29
23	54.369774	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3? Tell 172.31.0.4
24	54.369926	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0
25	54.370108	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.2
26	54.370128	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
27	63.879961	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
28	63.880210	172.31.0.4		28951 172.31.0.3		5001 UDP	59	28951 → 5001 Len=17

```
> Frame 8: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Xensource_3d:17:94 (00:16:3e:3d:17:94), Dst: Xensource_ae:c3:fd (00:16:3e:ae:c3:fd)
> Internet Protocol Version 4, Src: 172.31.0.4, Dst: 172.31.0.2
> User Datagram Protocol, Src Port: 5001, Dst Port: 6001
> Data (28 bytes)
```

0000	00 16 3e c3 fd 00 16 3e 3d 17 94 08 00 45 00	..>.... >.... E-
0010	00 38 fd 26 40 00 40 11 e5 49 ac 1f 00 04 ac 1f	8 @ @ I
0020	00 02 13 89 17 71 00 24 58 7a 63 68 61 74 5f 53	....q \$ Xchat_S
0030	54 41 52 54 5f 53 53 4c 5f 4e 4f 54 5f 53 55 50	TART_SSL_NOT_SUPPORTED
0040	50 4f 52 54 45 44	PORTED

- We can observe in the pcap that trudy1 sends 'chat\_START\_SSL\_NOT\_SUPPORTED' message to alice1, hence performing the 'Downgrade attack'.
- Trudy can observe all the chat messages communicated between alice1 and bob1, since they are sent in plain text.

trudy1\_task3.pcap

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
7	0.001071	172.31.0.3		5001 172.31.0.4		28951 UDP	62	5001 → 28951 Len=20
8	0.001145	172.31.0.4		5001 172.31.0.2		6001 UDP	70	5001 → 6001 Len=28
9	5.217855	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	Who has 172.31.0.2? Tell 172.31.0.4
10	5.217927	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.2
11	5.217934	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3? Tell 172.31.0.4
12	5.217948	Xensource_d2:a2:f0		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
13	5.218389	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.3
14	5.218422	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
15	5.218441	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	172.31.0.2 is at 00:16:3e:ae:c3:fd
16	5.218446	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0
17	15.144285	172.31.0.2		6001 172.31.0.4		5001 UDP	63	6001 → 5001 Len=21
18	15.144529	172.31.0.4		28951 172.31.0.3		5001 UDP	63	28951 → 5001 Len=21
19	32.025508	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
20	32.025769	172.31.0.4		5001 172.31.0.2		6001 UDP	59	5001 → 6001 Len=17
21	49.316315	172.31.0.2		6001 172.31.0.4		5001 UDP	71	6001 → 5001 Len=29
22	49.316553	172.31.0.4		28951 172.31.0.3		5001 UDP	71	28951 → 5001 Len=29
23	54.369774	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3? Tell 172.31.0.4
24	54.369926	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0
25	54.370108	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4? Tell 172.31.0.2
26	54.370128	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
27	63.879961	172.31.0.3		5001 172.31.0.4		28951 UDP	59	5001 → 28951 Len=17
28	63.880210	172.31.0.4		28951 172.31.0.3		5001 UDP	59	28951 → 5001 Len=17

```
> Frame 17: 63 bytes on wire (504 bits), 63 bytes captured (504 bits)
> Ethernet II, Src: Xensource_ae:c3:fd (00:16:3e:ae:c3:fd), Dst: Xensource_3d:17:94 (00:16:3e:3d:17:94)
> Internet Protocol Version 4, Src: 172.31.0.2, Dst: 172.31.0.4
> User Datagram Protocol, Src Port: 5001, Dst Port: 5001
> Data (21 bytes)
```

0000	00 16 3e c3 fd 00 16 3e ae c3 fd 08 00 45 00	..>.... >.... E-
0010	00 31 17 6c 40 00 40 11 cb 0b ac 1f 00 02 ac 1f	-1@ @ ..
0020	00 04 17 71 13 89 00 1d 58 73 48 69 20 62 6f 62	q \$ Xsh bob
0030	31 2c 20 68 6f 77 20 61 72 65 20 79 6f 75 3f	1, How a're you?

### Code snippet used by trudy1 to perform downgrade attack:

```
C secure_chat_interceptor.c X
Task_3 > C secure_chat_interceptor.c > eavesdropChat(char*,char*)
99 void eavesdropChat(char* aliceHostname, char* bobHostname) // (char* host1 = alice1, char* host2 = bob1)
191     // 7. Sending 'chat_START_SSL_NOT_SUPPORTED' message to alice1 from trudy1
192     sendto(trudy_alice_socket, (const char *)chat_START_SSL_NOT_SUPPORTED, strlen(chat_START_SSL_NOT_SUPPORTED), 0, (const struct sockaddr *)&alice_addr, sizeof(alice_addr));
193     printf("Msg sent to %s (%s:%d): %s\n", aliceHostname, inet_ntoa(alice_addr.sin_addr), ALICE_PORT, chat_START_SSL_NOT_SUPPORTED);
194 }
```

### Code snippet used by client (alice1) to start unsecure chat:

```
C secure_chat_app.c X
Task_4 > C secure_chat_app.c > runClient(char*)
654 void runClient(char *serverHostName)
695     // exchange_client_messages(): returns 1 if unsecure chatting is to be performed, 0 for secure chatting
696     if (exchange_client_messages(client_socket, server_addr) == 1){
697         perform_clientside_unsecure_chatting(client_socket, server_addr.s4);
698         // return 0;
699         exit(1);
700     }
```

## **TASK 4: Active MITM attack**

a) Answer:

### i) Trudy generating fake certificate for alice1:

## **Generating alice1's private key and CSR:**

### Int CA verifying the authenticity of alice1's CSR:

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014$ openssl req -in fake_alice.csr -noout -text | openssl dgst -md5 -out fake_alice_csr_digest_generated  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014$ openssl req -in fake_alice.csr -pubkey -noout > fake_alice_public_key.pem  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014$ openssl pkeyutl -verify -sigfile fake_alice_csr_sign -in fake_alice_csr_digest_generated -inkey fake_alice_public_key.pem -p  
ubin  
Signature Verified Successfully
```

## Int CA generating ‘fake crt’ for alice1:

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl x509 -req -in fake_alice.csr -CA int.crt -extfile fake_alice.txt -CAkey int_private_key.pem -CAcreateserial -out fake_alice.crt -days 730 -sha256
Certificate request self-signature ok
subject=CN = FakeAlice1.com, emailAddress = FakeAlice1@gmail.com, O = FakeAlice1, L = Nagpur, C = IN
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl dgst -md5 > fake_alice_crt_digest
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl pkeyutl -sign -in fake_alice_crt_digest -out fake_alice_crt_sign -inkey int_private_key.pem
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl x509 -in fakealice.crt -outform DER | openssl dgst -md5 > fake_alice_crt_digest
```

### Alice1 verifying the authenticity of ‘fake crt’:

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014_Fake_alice$ openssl x509 -in fakealice.crt -outform DER | openssl dgst -md5 > fakealice_crt_digest_generated  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014_Fake_alice$ openssl pkcs12 -verify -sigfile fake_alice_crt_sign -in fakealice_crt_digest_generated -inkey int_public_key.pem -pubin  
Signature Verified Successfully
```

Alice1 verifying the fake certificate:

```
mayuresh@mayuresh-HP-Laptop: ~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_alice
mayuresh@mayuresh-HP-Laptop: ~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_alice$ openssl verify -CAfile root.crt -untrusted int.crt fakealice.crt
fakealice.crt: OK
mayuresh@mayuresh-HP-Laptop: ~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_alice$
```

## **ii) Trudy generating fake certificate for bob1:**

### **Generating bob1's private key and CSR:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl ecparam -name prime256v1 -genkey -noout -out fake_bob_private_key.pem  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl req -new -config fake_bob.cnf -key fake_bob_private_key.pem -out fake_bob.csr  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl req -in fake_bob.csr -noout -text | openssl dgst -md5 -out fake_bob_csr_digest  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl pkeyutl -sign -in fake_bob_csr_digest -out fake_bob_csr_sign -inkey fake_bob_private_key.pem
```

### **Int CA verifying the authenticity of bob1's CSR:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl req -in fake_bob.csr -noout -text | openssl dgst -md5 -out fake_bob_csr_digest_generated  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl req -in fake_bob.csr -pubkey -noout > fake_bob_public_key.pem  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl pkeyutl -verify -sigfile fake_bob_csr_sign -in fake_bob_csr_digest_generated -inkey fake_bob_public_key.pem -pubin  
Signature Verified Successfully
```

### **Int CA generating 'fake crt' for bob1:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl x509 -req -in fake_bob.csr -CA int.crt -extfile fake_bob.txt -CAcreateserial -out fakebob.crt -days 730 -sha256  
Certificate request self-signature ok  
subject=CN = FakeBob1.com, emailAddress = FakeBob1@gmail.com, O = FakeBob1, L = Kolkata, C = IN  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl x509 -in fakebob.crt -outform DER | openssl dgst -md5 > fake_bob_crt_digest  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$ openssl pkeyutl -sign -in fake_bob_crt_digest -out fake_bob_crt_sign -inkey int_private_key.pem  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/int$
```

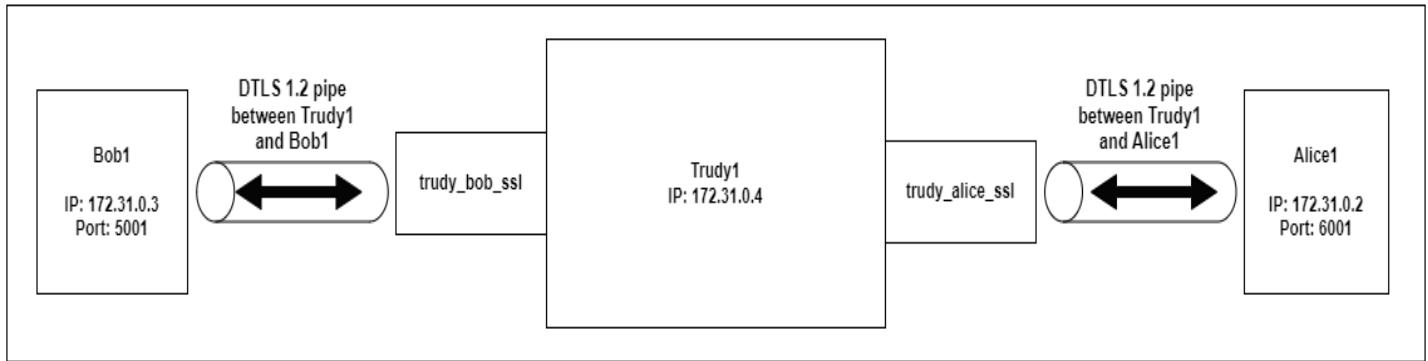
### **Bob1 verifying the authenticity of 'fake crt'**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl x509 -in fakebob.crt -outform DER | openssl dgst -md5 > fake_bob_crt_digest_generated  
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl verify -sigfile fake_bob_crt_sign -in fake_bob_crt_digest_generated -inkey int_public_key.pem -pubin  
Signature Verified Successfully
```

### **Bob1 verifying the fake certificate:**

```
mayuresh@mayuresh-HP-Laptop:~/Desktop/Secure_Chat_Tas4/CS23MTECH14003_CS23MTECH14007_CS23MTECH14014/Fake_bob$ openssl verify -CAfile root.crt -untrusted int.crt fakebob.crt  
fakebob.crt: OK
```

### **Active MITM attack overview:**



**Fig: DTLS v1.2 pipes in active MITM attack**

- Here, trudy1 performs an Active MITM attack by intercepting and modifying the chat messages communicated between alice1 and bob1.
- Trudy performs DNS poisoning by replacing the IP addresses of alice1 with bob1 and vice versa in the /etc/hosts file.
- Trudy1 forward the control messages exchanged between alice1 and bob1 without any tampering.
- Trudy1 establishes two DTLS v1.2 pipes:
  1. Between trudy1 and alice1
  2. Between trudy1 and bob1
- During DTLS v1.2 handshake with alice1, trudy1 sends the bob1's fake certificate to alice1. And, during DTLS v1.2 handshake with bob1, trudy1 sends the alice1's fake certificate to bob1.
- Therefore, trudy1 can easily see as well as tamper the encrypted chat messages exchanged between alice1 and bob1.
- In this case, trudy tampers the messages by appending '**\_TAMPERED\_BY\_TRUDY**' text to each chat message exchanged between alice1 and bob1.

### **Active MITM attack on secure chat output:**

```
File Edit Selection View Go Run Terminal Help ← → 🔍 ubuntu [SSH: 10.200.32.196] 🛡️ 🛡️ 🛡️ 🛡️

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

root@alice1:~# ./secure_chat_app -c bob1
Current Client Info:
Client IP: 172.31.0.2
Client Port: 6001
-----
Client socket created Successfully
-----
Client will chat with server:
Server Hostname: bob1
Server IP (after DNS poisoning): 172.31.0.4
Server Port: 5001
-----
Msg sent to Server: chat_hello
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_ACK
-----
Doing connection to trudy..... DTL
S 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
-----
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.co
m/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
-----
Depth:0, Cert Info: /CN=FakeBob1.com/emailAddress=FakeBob1@gma
il.com/0=FakeBob1/L=Kolkata/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
-----
Client connected to Server using DTLS
*****
Client has started chatting with Server *****

root@trudy1:~# ./secure_chat_active_interceptor -m alice1 bob1
*****
Started new MITM session *****
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4
-----
'trudy_alice_socket' socket created Successfully
-----
Waiting for 'chat_hello' msg from Alice1....
Msg recv from alice1 (172.31.0.2:28951): chat_hello
-----
'trudy_bob_socket' socket created Successfully
-----
Msg sent to bob1 (172.31.0.3:5001): chat_hello
Msg recv from bob1 (172.31.0.3:5001): chat_ok_reply
Msg sent to alice1 (172.31.0.2:28951): chat_ok_reply
Msg recv from alice1 (172.31.0.2:28951): chat_START_SSL
Msg sent to bob1 (172.31.0.3:5001): chat_START_SSL
Msg recv from bob1 (172.31.0.3:5001): chat_START_SSL_ACK
Msg sent to alice1 (172.31.0.2:28951): chat_START_SSL_ACK
-----
*** ALL CONTROL MESSAGES ARE COMMUNICATED CORRECTLY ***
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
-----
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.co
m/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
-----
Depth:0, Cert Info: /CN=FakeAlice1.com/emailAddress=FakeAlice1@gmail.com/0=FakeAlice1/L=Nagpur/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
-----
Server connected to Client using DTLS 1.2
*****
Server has started chatting with Client *****

root@bob1:~# ./secure_chat_app -s
Server Info:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for new client connection....
Connected to client: '172.31.0.4'
Msg recv from Client (172.31.0.4:6001): chat_hello
Msg sent to Client (172.31.0.4:6001): chat_ok_reply
Msg recv from Client (172.31.0.4:6001): chat_START_SSL
Msg sent to Client (172.31.0.4:6001): chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
-----
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.co
m/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
-----
Depth:0, Cert Info: /CN=FakeAlice1.com/emailAddress=FakeAlice1@gmail.com/0=FakeAlice1/L=Nagpur/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
-----
Server connected to Client using DTLS 1.2
*****
Server has started chatting with Client *****

From Client: Hi bob1_TAMPERED_BY_TRUDY
Enter message for Client: Hi alice1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
S 1.2 Handshake started ....  
.... Certificate Verification Started ....  
  
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN  
Certificate having depth 2 is NOT expired  
  
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN  
Certificate having depth 1 is NOT expired  
  
Depth:0, Cert Info: /CN=FakeBob1.com/emailAddress=FakeBob1@gmail.com/O=FakeBob1/L=Kolkata/C=IN  
Certificate having depth 0 is NOT expired  
  
.... Certificate Verification Ended ....  
  
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384  
.... DTLS 1.2 Handshake ended ....  
  
Client connected to Server using DTLS  
***** Client has started chatting with Server *****  
  
Enter message for Server: Hi bob1  
  
From Server: Hi alice1_TAMPERED_BY_TRUDY  
Enter message for Server: Can we go for a movie?  
  
From Server: sure. Which movie?_TAMPERED_BY_TRUDY  
Enter message for Server: Titanic?  
  
From Server: ok, sure_TAMPERED_BY_TRUDY  
Enter message for Server: See you!  
  
From Server: bye_TAMPERED_BY_TRUDY  
Enter message for Server: chat_close  
  
***** Chat closed *****  
-----  
root@alice1:~#
```

```
Certificate having depth 1 is NOT expired  
  
Depth:0, Cert Info: /CN=Bob1.com/emailAddress=Bob1@gmail.com/O=Bob1/L=Kolkata/C=IN  
Certificate having depth 0 is NOT expired  
.... Certificate Verification Ended ....  
  
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384  
.... DTLS 1.2 Handshake ended ....  
  
Client connected to Server using DTLS  
-----  
---- Successfully created trudy <-> bob DTLS pipe -----  
***** Active MITM attack started *****  
  
From alice1: Hi bob1  
From bob1: Hi alice1  
  
From alice1: Can we go for a movie?  
From bob1: sure. Which movie?  
  
From alice1: Titanic?  
From bob1: ok, sure  
  
From alice1: See you!  
From bob1: bye  
  
From alice1: chat_close  
***** Chat closed *****  
  
***** Started new MITM session *****  
Alice's IP: 172.31.0.2  
Bob's IP: 172.31.0.3  
Trudy's IP: 172.31.0.4  
  
'trudy_alice_socket' socket created Successfully  
-----  
Waiting for 'chat_hello' msg from Alice1....  
|
```

```
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/O=iTS ROOT R1/L=Hyderabad/C=IN  
Certificate having depth 2 is NOT expired  
  
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/O=iTS CA 1R3/L=Nashik/C=IN  
Certificate having depth 1 is NOT expired  
  
.... Certificate Verification Ended ....  
  
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384  
.... DTLS 1.2 Handshake ended ....  
  
Server connected to Client using DTLS 1.2  
***** Server has started chatting with Client *****  
  
From Client: Hi bob1_TAMPERED_BY_TRUDY  
Enter message for Client: Hi alice1  
  
From Client: Can we go for a movie?_TAMPERED_BY_TRUDY  
Enter message for Client: sure. Which movie?  
  
From Client: Titanic?_TAMPERED_BY_TRUDY  
Enter message for Client: ok, sure  
  
From Client: See you!_TAMPERED_BY_TRUDY  
Enter message for Client: bye  
  
From Client: chat_close  
***** Chat closed *****  
  
-----  
Server socket created successfully  
-----  
Server started listening for new client connection....  
|
```

## 1. Tcpdump trace captured at 'alice1':

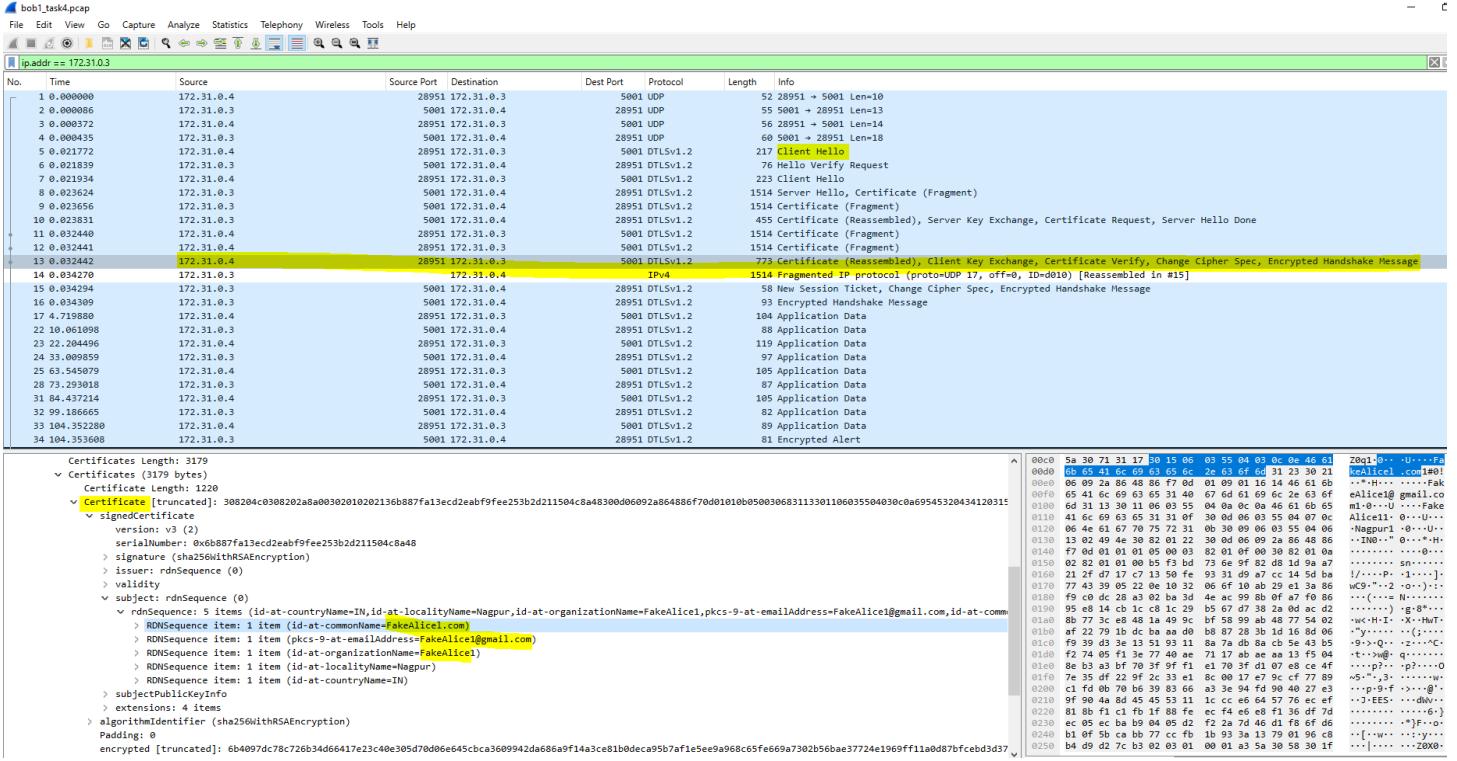
- In the below pcap trace, we can observe that the ‘Client Hello’ message is sent from alice1 (172.31.0.2) to trudy1 (172.31.0.4) during DTLS v1.2 handshake.
  - We can see that trudy1 has sent a **‘fake certificate’** of bob1 to alice1.

The screenshot shows a network capture in Wireshark. The timeline pane at the top lists 192 captured frames, with frame 10 highlighted in yellow. The details pane below shows the reassembled certificate message from Bob, which includes fields like Version, Type, and Subject. The bytes pane on the right shows the raw hex and ASCII data for the selected frame.

- From below pcap trace, we can observe that the chat messages between trudy1 and alice1 are encrypted using DTLS v1.2.

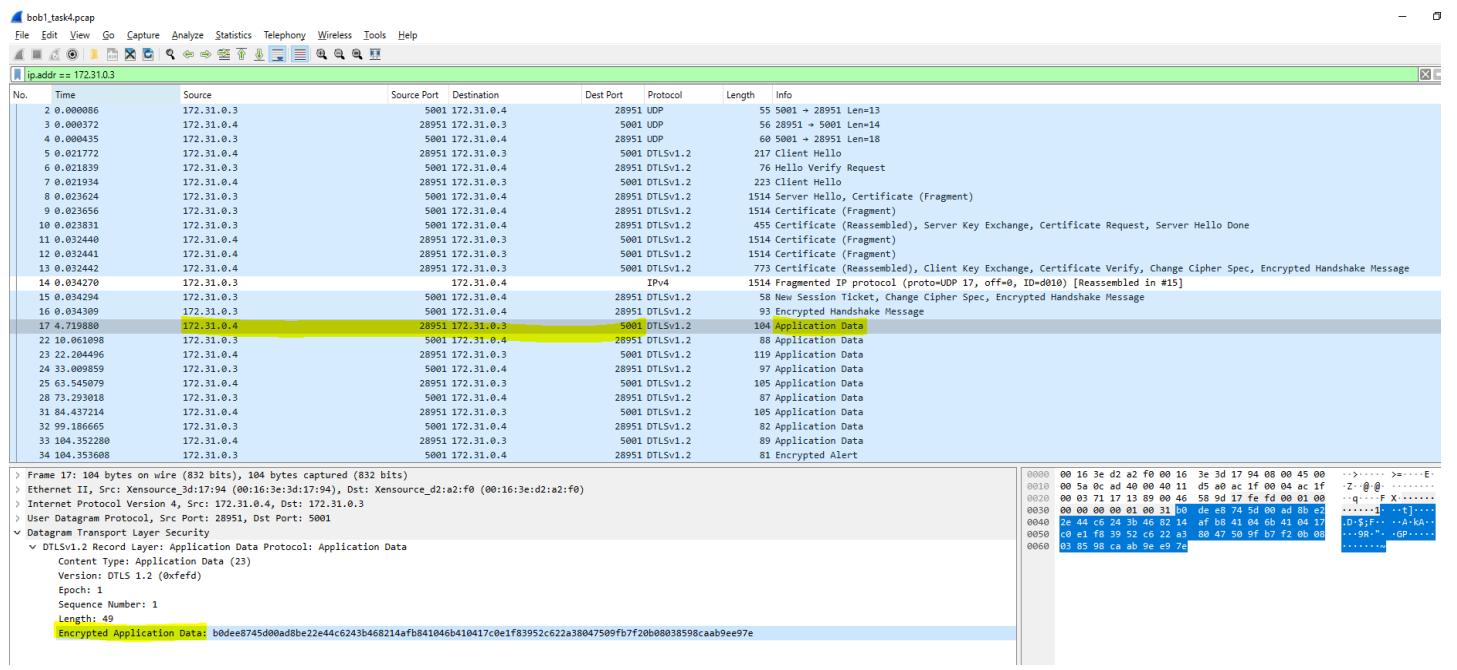
## 2. Tcpdump trace captured at 'bob1':

- In the pcap below, we can observe that the 'Client Hello' message is sent from trudy1 (172.31.0.4) to bob1 (172.31.0.3) during DTLS v1.2 handshake.
- We can see that trudy1 has sent a '**fake certificate**' of alice1 to bob1.



0x00	5a 30 71 31 17	30 15 06 03 55 04 03 0c 0e 46 61	Z@q1@...-U...-Fa
0x0d	6b 65 41 6c 09 63 6c 6c 6c 6c 6c 6c	0d 09 01 16 40 40 40 40 40 40 40 40	keAlice1.co...#01
0x10	65 41 6c 69 63 65 31 40	67 64 61 69 6c 2e 63 6f	eAlice1@gmail.co
0x11	41 6c 69 63 65 31 31 0f	30 0d 06 03 55 04 07 0c	Alice1...-U...-U
0x12	06 4e 61 67 70 75 72 31	0b 09 09 06 03 55 04 06	-Nagpu1...-U...-U
0x13	13 02 61 67 70 75 72 31	02 0d 04 06 03 55 04 06	...IND...-U...-U
0x14	65 41 6c 69 63 65 31 40	67 64 61 69 6c 2e 63 6f	...IND...-U...-U
0x15	02 82 01 01 09 b5 f3 bd	73 6e 97 82 d8 1d 9a a7	.....sn.....
0x16	21 2f d7 17 c7 13 50 fd	93 31 d9 a7 cc 14 5d ba	[/...-P...-1...]-
0x17	77 43 39 05 22 0e 10 32	06 6f 10 ab 29 e1 3a 86	wC9...-2 o...)-
0x18	f9 c0 dc 2b 0d 0b 0b 0b	ac 99 8b 0f 0f 0f 0f 0f	...((...-P...-1...)-
0x19	01 0b 01 0b 01 0b 01 0b	29 0d 0d 0d 0d 0d 0d 0d	g*8...
0x1a	8b 77 3c c8 1a 49 9c	b6 59 9b 48 77 54 02	...wC-H...-X-HwT...
0x1b	a2 22 79 1b dc ba aa	b6 87 28 3b 1d 16 8d 06	...y...>...;
0x1c	f9 39 d3 3e 13 51 93 11	8a 7a db 8a cb 5e 43 b5	>>>Q...>...-C
0x1d	f2 74 05 f1 3e 77 40 ae	71 17 ab ae a3 19 f5 04	-t...-W... q...-o...
0x1e	ec 05 05 05 05 05 05 05	11 11 11 11 11 11 11 11	p...-o...-o...-o...
0x1f	7e 35 df 22 9f 2c 33 e1	8c 00 17 e7 9c cf 77 89	=>...-3...-y...-y...
0x20	c1 fd 0b 70 b6 39 83 6d	a3 3e 94 fd 98 40 27 e3	...p...-9-f ...>...-B...
0x21	9f 90 4a fd 45 45 53 11	11 cc ee 64 57 76 ec ef	-J-EES...-div...
0x22	81 8b f1 f1 b8 ff	cc f4 e6 8f 11 36 df 7d	.....-E...-E...
0x23	ec 05 05 05 b9 09 05 05	2a 74 40 01 01 01 01 01	...J...-o...-o...
0x24	5d 5a 5a 5a 5a 5a 5a 5a	1b 1b 34 34 34 34 34 34	...-y...-y...-y...
0x25	b4 d9 d2 7c b3 02 03 01	09 01 a3 a3 3a 38 38 38 0f	...-Z...-Z...-Z...

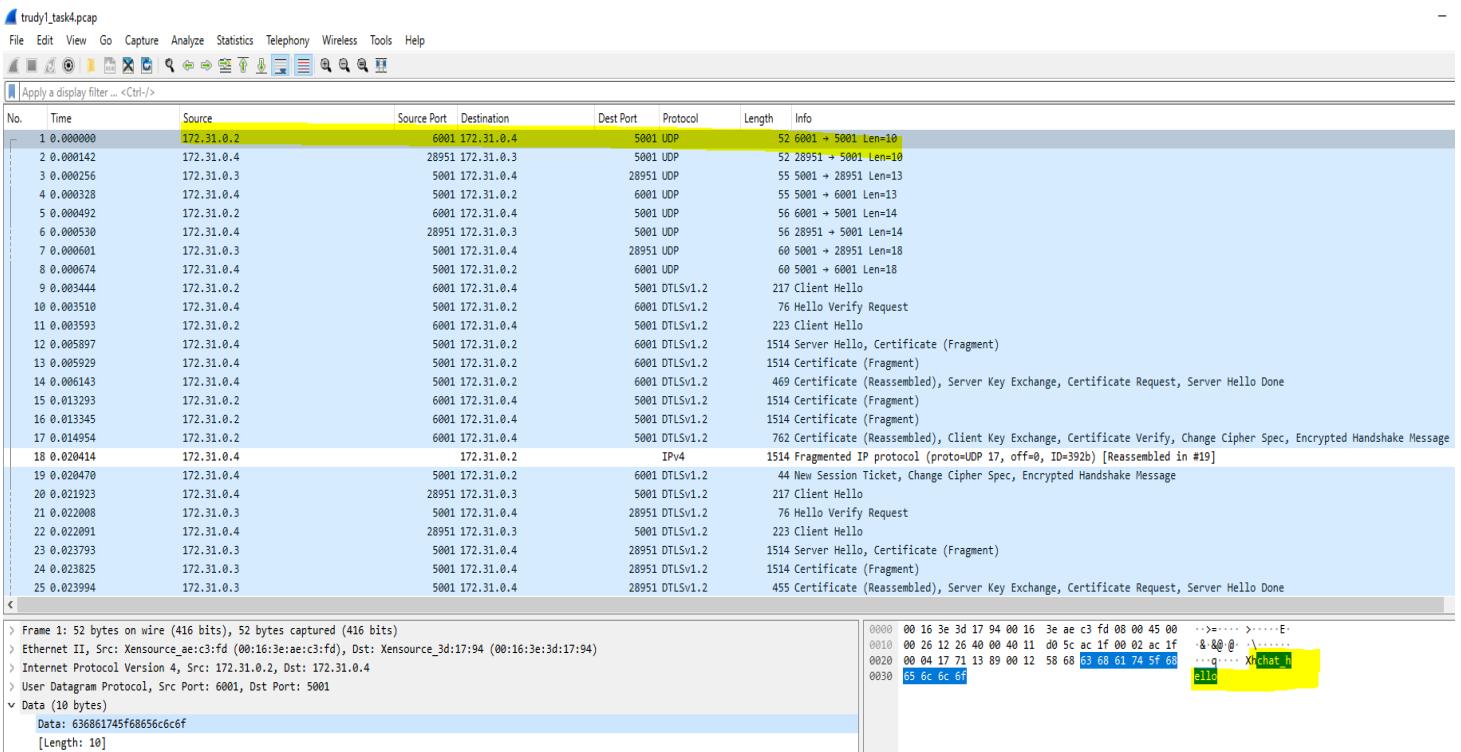
- From below pcap trace, we can observe that the chat messages between trudy1 and bob1 are encrypted using DTLS v1.2.



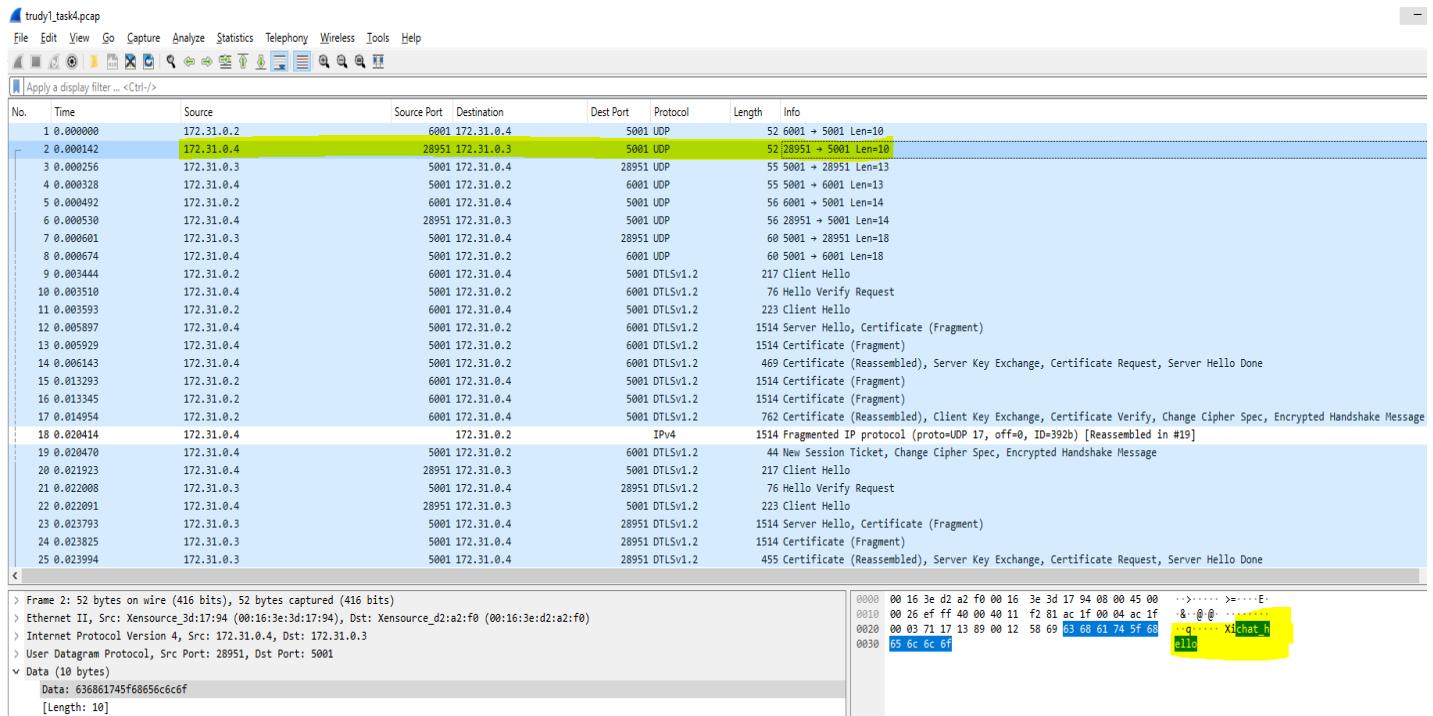
0x00	00 16 3e d2 a2 f0 00 16	3e 3d 17 94 08 00 45 00	>>>...>...-E...
0x10	00 5a 0c ad 40 00 40 11	5d a0 ac 1f 00 04 ac 1f	Z...@...-F...
0x20	00 03 71 17 13 89 00 46	58 9d 17 fe fd 00 01 00	...q...-F X...-....
0x30	00 00 00 00 01 00 31 b0	b6 e8 74 5d 00 ad 8b e2	.....-t]...-t]...-t]...
0x40	0e 44 c6 24 3b 46 62 14	aF b8 a1 04 6b 41 04 17	-D-\$!...-A-EA...
0x50	e1 f8 39 52 c6 a3	80 47 5b 9f 07 f2 00 08	...-9R...-D-P...-...
0x60	03 09 c8 00 49 7c	03 09 c8 00 49 7c	.....

### 3. Tcpdump trace captured at 'trudy1':

- In the pcap below, we can observe that trudy1 (172.31.0.4) receives the 'chat\_hello' plain text control message from alice1 (172.31.0.2).

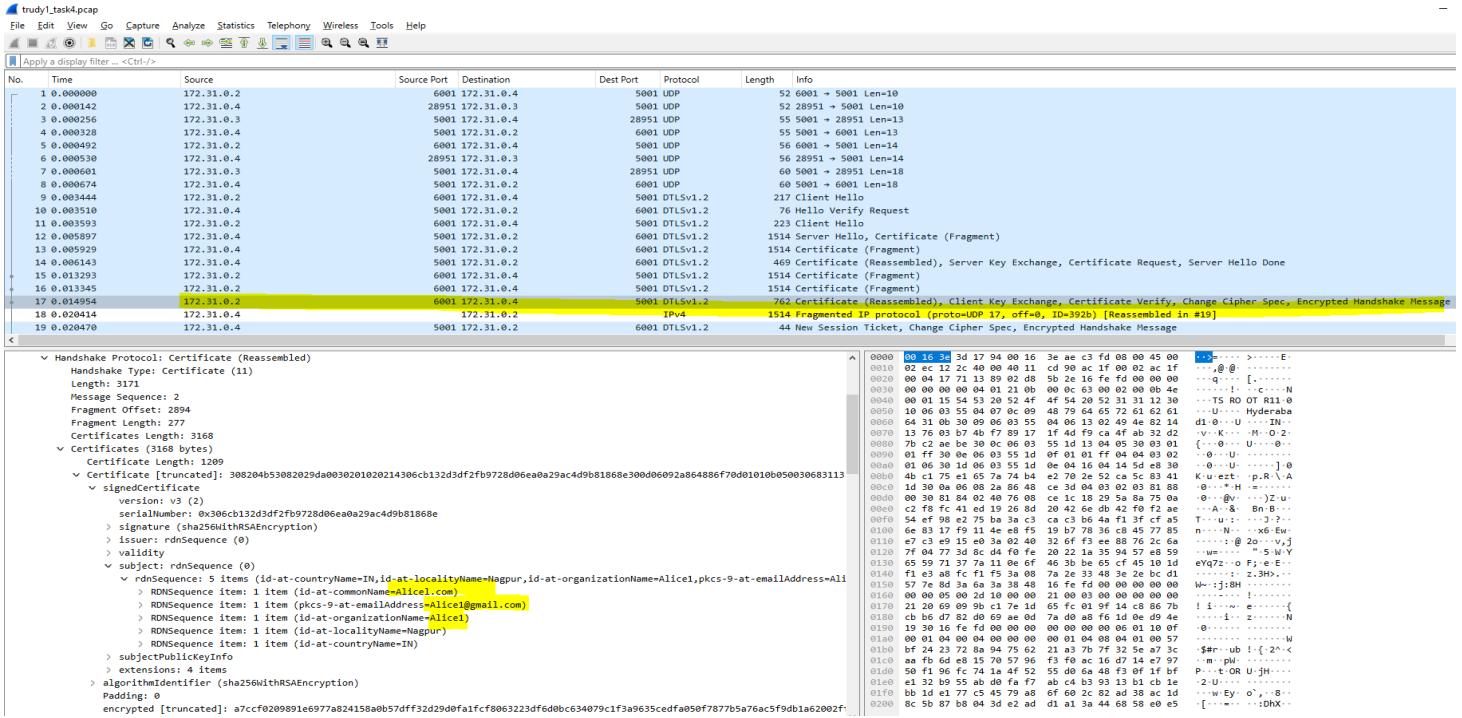


- Trudy1 (172.31.0.4) forwards this 'chat\_hello' plain text control message to bob1 (172.31.0.3).
- Similarly, trudy1 forwards all the control messages sent from alice1 to bob1 and vice versa.



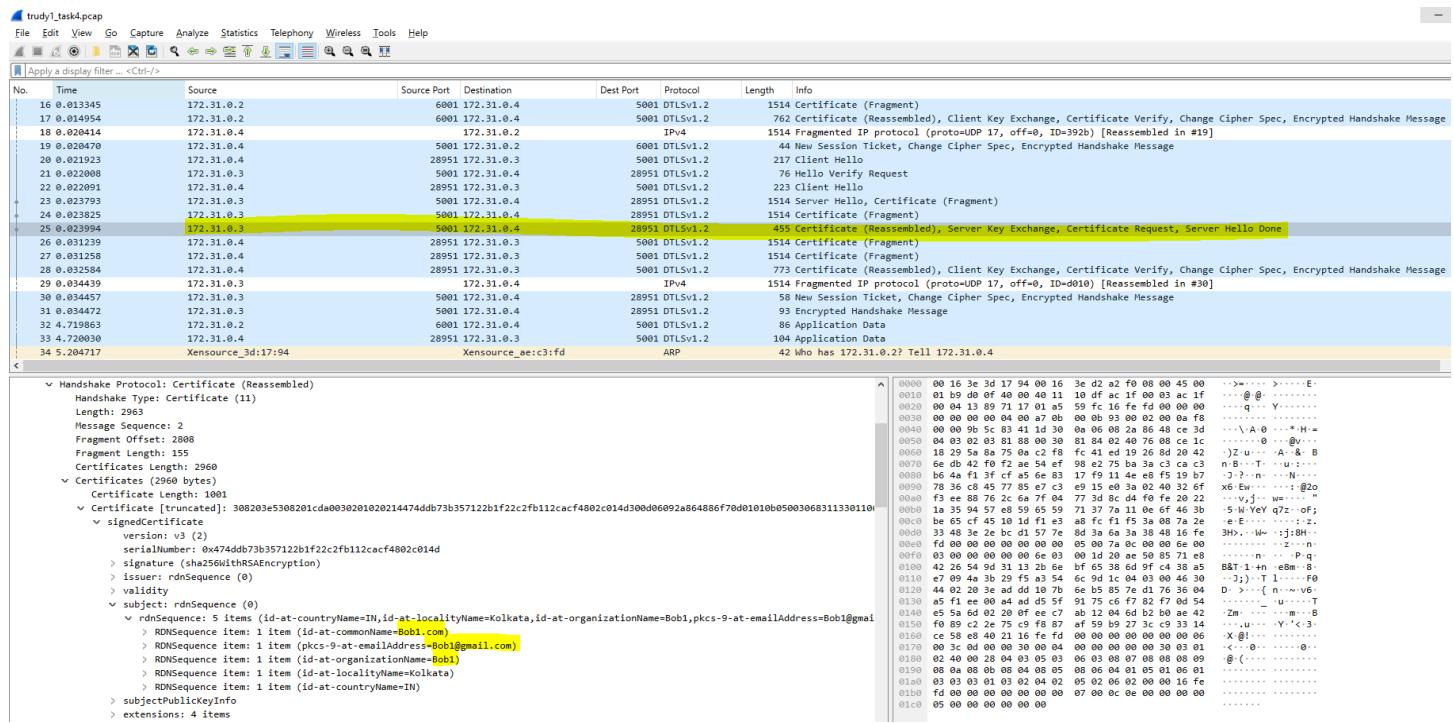
## Certificate received by trudy1 from alice1:

- Alice1 (172.31.0.2) shares her genuine certificate to trudy1 (172.31.0.4) during DTLS v1.2 handshake.



## Certificate received by trudy1 from bob1:

- Bob1 (172.31.0.3) shares her genuine certificate to trudy1 (172.31.0.4) during DTLS v1.2 handshake.



**Code used to establish two DTLS pipes:**

```
C secure_chat_active_interceptor.c X  
Task_4 > C secure_chat_active_interceptor.c > eavesdropChat(char *, char *)  
404 ✓ void eavesdropChat(char* aliceHostname, char* bobHostname) // (char* host1 = alice1, char* host2 = bob1)  
504     // Establishing the trudy <-> alice DTLS pipe  
505     SSL *trudy_alice_ssl;  
506     trudy_alice_ssl = create_trudy_to_alice_DTLS_pipe(trudy_alice_socket);  
507  
508     // Establishing the trudy <-> bob DTLS pipe  
509     SSL *trudy_bob_ssl;  
510     trudy_bob_ssl = create_trudy_to_bob_DTLS_pipe(trudy_bob_socket, BOB_IP);
```

Two DTLS v1.2 pipes are established:

1. Between trudy1 and alice1 indicated by ‘trudy\_alice\_ssl’ in the code snippet
2. Between trudy1 and bob1 indicated by ‘trudy\_bob\_ssl’ in the code snippet

## Active MITM attack performed by trudy1:

- In the below output screenshot, we can observe that, the trudy1 is performing the Active MITM attack by tampering the chat messages communicated between alice1 and bob1.
- Here, trudy1 is tampering the chat messages by appending “**\_TAMPERED\_BY\_TRUDY**” text to them.

The screenshot shows three terminal windows illustrating the Active MITM attack. The left window (Alice1) shows a conversation with Bob1. The middle window (Trudy1) shows it intercepting and modifying the messages. The right window (Bob1) shows the final tampered messages received.

```

.... DTLS 1.2 Handshake ended ....
-----
Client connected to Server using DTLS
-----
***** Client has started chatting with Server *****
Enter message for Server: Hi bob1
From Server: Hi alice1_TAMPERED_BY_TRUDY
Enter message for Server: How are you?
From Server: I am fine_TAMPERED_BY_TRUDY
Enter message for Server: nice
From Server: chat_close
***** Chat closed *****

-----
root@alice1:~#
root@alice1:~#
root@alice1:~#
root@alice1:~#
root@alice1:~#
root@alice1:~# []

----- Successfully created trudy <-> bob DTLS pipe -----
***** Active MITM attack started *****
From alice1: Hi bob1
From bob1: Hi alice1

From alice1: How are you?
From bob1: I am fine

From alice1: nice
From bob1: chat_close
***** Chat closed *****

----- Started new MITM session -----
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4

----- 'trudy_alice_socket' socket created Successfully -----
----- Waiting for 'chat_hello' msg from Alice1.... []

----- HA384 -----
.... DTLS 1.2 Handshake ended ....
-----
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****
From Client: Hi bob1_TAMPERED_BY_TRUDY
Enter message for Client: Hi alice1

From Client: How are you?_TAMPERED_BY_TRUDY
Enter message for Client: I am fine

From Client: nice_TAMPERED_BY_TRUDY
Enter message for Client: chat_close
***** Chat closed *****

-----
Server socket created successfully
-----
Server started listening for new client connection...

```

## Code snippet used for msg tampering:

The code snippet shows the implementation of the eavesdropChat function, which performs an Active MITM attack by tampering with messages sent between Alice1 and Bob1.

```

C secure_chat_active_interceptor.c X
Task_4 > C secure_chat_active_interceptor.c > eavesdropChat(char *, char *)
404 void eavesdropChat(char* aliceHostname, char* bobHostname) // (char* host1 = alice1, char* host2 = bob1)
513     printf("\n***** Active MITM attack started *****\n");
514     int len;
515     while(1)
516     {
517         // -----
518         memset(buffer, 0, sizeof(buffer));
519         do
520         {
521             len = SSL_read(trudy_alice_ssl, buffer, sizeof(buffer));
522         } while (len == -1);
523         if (len > 0)
524         {
525             printf("\nFrom alice1: %s", buffer);
526             if (strcmp(buffer, "chat_close") != 0) {
527                 strcat(buffer, "_TAMPERED_BY_TRUDY");
528             }
529         }
530     }
531     // write the modified msg to trudy_bob_ssl
532     len = SSL_write(trudy_bob_ssl, buffer, strlen(buffer));
533     if (len > 0)
534     {
535         if (strcmp(buffer, "chat_close") == 0)
536         {
537             printf("\n***** Chat closed *****\n");
538             break;
539         }
540     }

```

## Chat close scenarios:

## 1. When client (alice1) closes the chat:

```
.... Certificate Verification Ended ....  
  
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384  
  
.... DTLS 1.2 Handshake ended ....  
  
-----  
Client connected to Server using DTLS  
  
***** Client has started chatting with Server *****  
  
Enter message for Server: Hi bob1  
  
From Server: Hi alice1_TAMPERED_BY_TRUDY  
Enter message for Server: Can we go for a movie?  
  
From Server: sure. Which movie?_TAMPERED_BY_TRUDY  
Enter message for Server: Titanic?  
  
From Server: ok, sure_TAMPERED_BY_TRUDY  
Enter message for Server: See you!  
  
From Server: bye_TAMPERED_BY_TRUDY  
Enter message for Server: chat_close  
  
***** Chat closed *****  
  
-----  
root@alice1:~#
```

```
***** Active MITM attack started *****

From alice1: Hi bob1
From bob1: Hi alice1

From alice1: Can we go for a movie?
From bob1: sure. Which movie?

From alice1: Titanic?
From bob1: ok, sure

From alice1: See you!
From bob1: bye

From alice1: chat_close
***** Chat closed *****
```

---

```
***** Started new MITM session *****
Alice's IP: 172.31.0.2
Bob's IP: 172.31.0.3
Trudy's IP: 172.31.0.4

'trudy_alice_socket' socket created Successfully

Waiting for 'chat_hello' msg from Alice1...
|
```

```
A384

.... DTLS 1.2 Handshake ended ....
-----
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****

From Client: Hi bob1_TAMPERED_BY_TRUDY
Enter message for Client: Hi alice1

From Client: Can we go for a movie?_TAMPERED_BY_TRUDY
Enter message for Client: sure. Which movie?

From Client: Titanic?_TAMPERED_BY_TRUDY
Enter message for Client: ok, sure

From Client: See you!_TAMPERED_BY_TRUDY
Enter message for Client: bye

From Client: chat_close
***** Chat closed *****

-----
Server socket created successfully
-----
Server started listening for new client connection...
|
```

## **2. When server (bob1) closes the chat:**

```
.... DTLS 1.2 Handshake ended ....  
-----  
Client connected to Server using DTLS  
-----  
***** Client has started chatting with Server *****  
  
Enter message for Server: Hi bob1  
  
From Server: Hi alice1_TAMPERED_BY_TRUDY  
Enter message for Server: How are you?  
  
From Server: I am fine_TAMPERED_BY_TRUDY  
Enter message for Server: nice  
  
From Server: chat_close  
***** Chat closed *****  
  
-----  
root@alice1:~#  
root@alice1:~#  
root@alice1:~#  
root@alice1:~#  
root@alice1:~#  
root@alice1:~#
```

```
----- Successfully created trudy <-> bob DTLS pipe -  
  
***** Active MITM attack started *****  
  
From alice1: Hi bob1  
From bob1: Hi alice1  
  
From alice1: How are you?  
From bob1: I am fine  
  
From alice1: nice  
From bob1: chat_close  
***** Chat closed *****  
  
-----  
***** Started new MITM session *****  
Alice's IP: 172.31.0.2  
Bob's IP: 172.31.0.3  
Trudy's IP: 172.31.0.4  
  
-----  
'trudy_alice_socket' socket created Successfully  
  
-----  
Waiting for 'chat_hello' msg from Alice1....  
|
```

HA384

---

.... DTLS 1.2 Handshake ended ....

---

-----

Server connected to Client using DTLS 1.2

---

-----

\*\*\*\*\* Server has started chatting with Client \*\*\*\*\*

-----

From Client: Hi bob1\_TAMPERED\_BY\_TRUDY

Enter message for Client: Hi alice1

-----

From Client: How are you?\_TAMPERED\_BY\_TRUDY

Enter message for Client: I am fine

-----

From Client: nice\_TAMPERED\_BY\_TRUDY

Enter message for Client: chat\_close

-----

\*\*\*\*\* Chat closed \*\*\*\*\*

---

-----

Server socket created successfully

---

-----

Server started listening for new client connection....

- In both scenarios, when the chat is closed, both the trudy1 and bob1 keeps on listening for new client connection requests.

### Injecting packet loss

Injecting packet loss at ‘alice1’:

```
root@alice1:~# sudo tc qdisc add dev eth0 root netem loss 18%
root@alice1:~# sudo tc qdisc show dev eth0
qdisc netem 800d: root refcnt 3 limit 1000 loss 18%
root@alice1:~# █
```

Injecting packet loss at ‘bob1’:

```
root@bob1:~# sudo tc qdisc add dev eth0 root netem loss 18%
root@bob1:~# sudo tc qdisc show dev eth0
qdisc netem 800e: root refcnt 3 limit 1000 loss 18%
root@bob1:~# █
```

Injecting packet loss at ‘trudy1’:

```
root@trudy1:~# sudo tc qdisc add dev eth0 root netem loss 18%
root@trudy1:~# sudo tc qdisc show dev eth0
qdisc netem 800f: root refcnt 3 limit 1000 loss 18%
root@trudy1:~# █
```

**Secure chat along with active MITM output after injecting packet loss of 18%:**

```
File Edit Selection View Go Run Terminal Help ← → ubuntu [SSH: 10.200.32.196]
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Client socket created Successfully
-----
Client will chat with server:
Server Hostname: bob1
Server IP (after DNS poisoning): 172.31.0.4
Server Port: 5001
-----
Msg sent to Server: chat_hello
Msg recv from Server: chat_ok_reply
Msg sent to Server: chat_START_SSL
Msg recv from Server: chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
Depth:0, Cert Info: /CN=FakeBob1.com/emailAddress=FakeBob1@gmail.com/0=FakeBob1/L=Kolkata/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
Client connected to Server using DTLS
-----
Successfully created trudy <-> bob DTLS pipe
***** Active MITM attack started *****
From alice1: Hi bob1
From bob1: Hi alice1
From alice1: How are you?
-----
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****
From Client: Hi bob1_TAMPERED_BY_TRUDY
Enter message for Client: Hi alice1
-----
Server Info:
Server Hostname: bob1
Server IP: 172.31.0.3
Server Port: 5001
-----
Server socket created successfully
-----
Server started listening for new client connection....
Connected to client: '172.31.0.4'
Msg recv from Client (172.31.0.4:6001): chat_hello
Msg sent to Client (172.31.0.4:6001): chat_ok_reply
Msg recv from Client (172.31.0.4:6001): chat_START_SSL
Msg sent to Client (172.31.0.4:6001): chat_START_SSL_ACK
-----
.... DTLS 1.2 Handshake started ....
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
Depth:0, Cert Info: /CN=Bob1.com/emailAddress=Bob1@gmail.com/0=Bob1/L=Kolkata/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
Client connected to Server using DTLS
-----
Successfully created trudy <-> bob DTLS pipe
***** Active MITM attack started *****
From alice1: Hi bob1
From bob1: Hi alice1
From alice1: How are you?
-----
.... Certificate Verification Started ....
Depth:2, Cert Info: /CN=iTS ROOT R1/emailAddress=rootCA@gmail.com/0=iTS ROOT R1/L=Hyderabad/C=IN
Certificate having depth 2 is NOT expired
Depth:1, Cert Info: /CN=iTS CA 1R3/emailAddress=intCA@gmail.com/0=iTS CA 1R3/L=Nashik/C=IN
Certificate having depth 1 is NOT expired
Depth:0, Cert Info: /CN=FakeAlice1.com/emailAddress=FakeAlice1@gmail.com/0=FakeAlice1/L=Nagpur/C=IN
Certificate having depth 0 is NOT expired
.... Certificate Verification Ended ....
Cipher suite selected by server: ECDHE-ECDSA-AES256-GCM-SHA384
.... DTLS 1.2 Handshake ended ....
Server connected to Client using DTLS 1.2
-----
***** Server has started chatting with Client *****
From Client: Hi bob1_TAMPERED_BY_TRUDY
Enter message for Client: Hi alice1
```

- In the above screenshot, we can observe that the chat message ‘How are you?’ sent from trudy1 to bob1 gets lost due to injected packet loss of 18%.

## Same thing can be confirmed from below tcpdump pcap traces:

### Tcpdump trace captured at alice1:

alice1\_pkt\_loss\_task4.pcap

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10
2	0.000080	172.31.0.4	5001	172.31.0.2	6001	UDP	55	5001 → 6001 Len=13
3	0.001468	172.31.0.2	6001	172.31.0.4	5001	UDP	56	6001 → 5001 Len=14
4	0.001723	172.31.0.4	5001	172.31.0.2	6001	UDP	60	5001 → 6001 Len=18
5	0.006343	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	217	Client Hello
6	0.006430	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	76	Hello Verify Request
7	0.006488	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	223	Client Hello
8	0.008340	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	1514	Server Hello, Certificate (Fragment)
9	0.008363	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	1514	Certificate (Fragment)
10	0.008814	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	468	Certificate (Reassembled), Server Key Exchange, Certificate Request, Server Hello Done
11	0.015339	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	1514	Certificate (Fragment)
12	0.015373	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	1514	Certificate (Fragment)
13	0.017228	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	762	Certificate (Reassembled), Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
14	0.021072	172.31.0.4		172.31.0.2		IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=844b) [Reassembled in #15]
15	0.021133	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	44	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
16	5.006729	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4 Tell 172.31.0.2
17	5.007026	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	Who has 172.31.0.2 Tell 172.31.0.4
18	5.007052	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	172.31.0.2 is at 00:16:3e:ae:c3:fd
19	5.007459	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
20	21.904997	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	86	Application Data → Hi bob1
21	29.765287	172.31.0.4	5001	172.31.0.2	6001	DTLSv1.2	106	Application Data → Hi alice1
22	36.603248	172.31.0.2	6001	172.31.0.4	5001	DTLSv1.2	91	Application Data → How are you?
23	41.614642	Xensource_ae:c3:fd		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4 Tell 172.31.0.2
24	41.614898	Xensource_3d:17:94		Xensource_ae:c3:fd		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94

```
> Frame 22: 91 bytes on wire (728 bits), 91 bytes captured (728 bits)
> Ethernet II, Src: Xensource_ae:c3:fd (00:16:3e:ae:c3:fd), Dst: Xensource_3d:17:94 (00:16:3e:3d:17:94)
> Internet Protocol Version 4, Src: 172.31.0.2, Dst: 172.31.0.4
> User Datagram Protocol, Src Port: 5001, Dst Port: 5001
└ DTLSv1.2 Record Layer: Application Data Protocol: Application Data
    Content Type: Application Data (23)
    Version: DTLS 1.2 (0xfeed)
    Epoch: 1
    Sequence Number: 2
    Length: 36
    Encrypted Application Data: 330e029fa59f4be68e0503504ac9b9b297ca093d458bcb6b485941c2e467b2803faf98b509
```

In the above screenshot, we can observe that the last chat message sent by alice1 (172.31.0.2) to trudy1 (172.31.0.4) is 'How are you?'.

### Tcpdump trace captured at bob1:

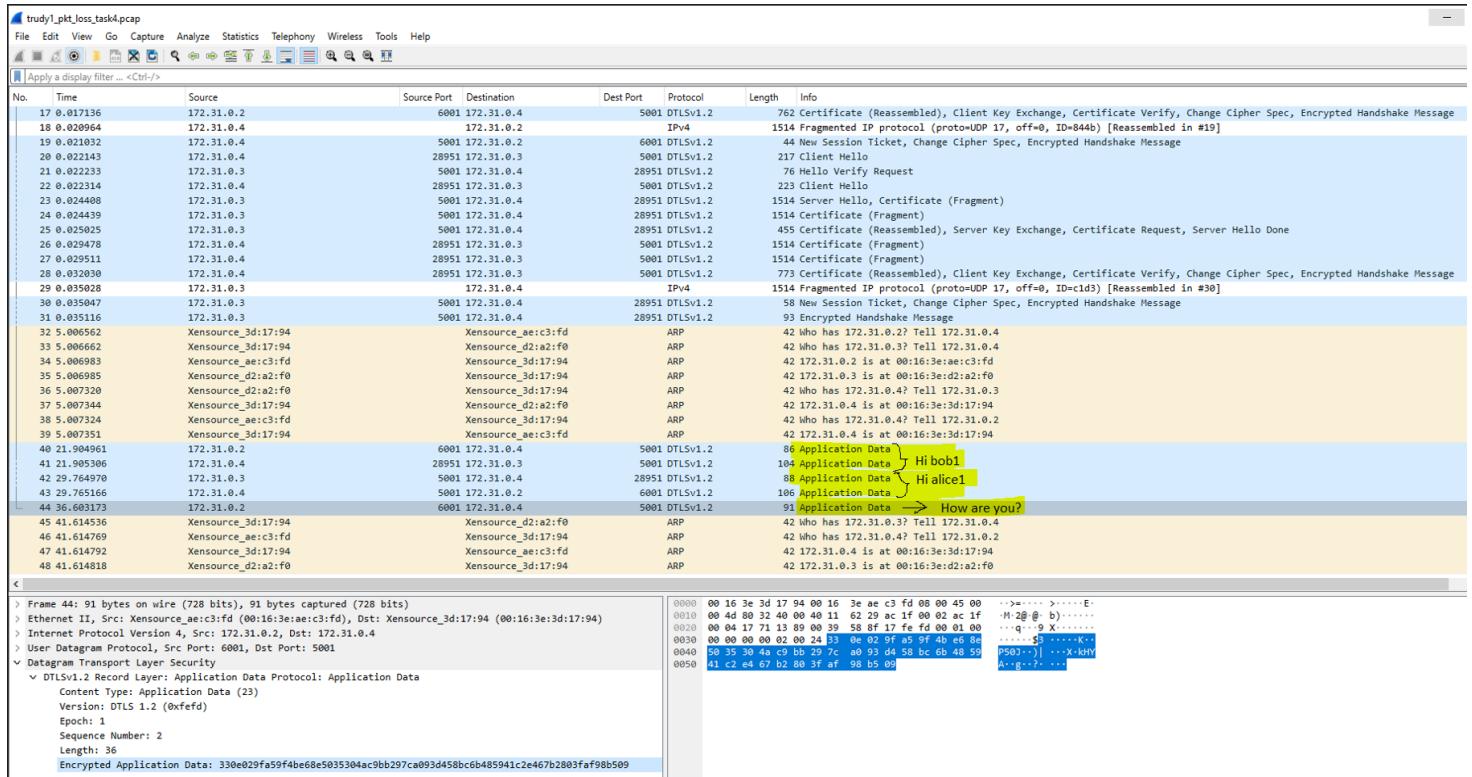
bob1\_pkt\_loss\_task4.pcap

No.	Time	Source	Source Port	Destination	Dest Port	Protocol	Length	Info
1	0.000000	172.31.0.2	6001	172.31.0.4	5001	UDP	52	6001 → 5001 Len=10
2	0.000333	172.31.0.4	28951	172.31.0.3	5001	UDP	52	28951 → 5001 Len=10
3	0.000576	172.31.0.3	5001	172.31.0.4	28951	UDP	55	5001 → 28951 Len=13
4	0.001438	172.31.0.4	28951	172.31.0.3	5001	UDP	56	28951 → 5001 Len=14
5	0.001544	172.31.0.3	5001	172.31.0.4	28951	UDP	60	5001 → 28951 Len=18
6	0.022145	172.31.0.4	28951	172.31.0.3	5001	DTLSv1.2	217	Client Hello
7	0.022219	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	76	Hello Verify Request
8	0.022313	172.31.0.4	28951	172.31.0.3	5001	DTLSv1.2	223	Client Hello
9	0.024394	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	1514	Server Hello, Certificate (Fragment)
10	0.024430	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	1514	Certificate (Fragment)
11	0.025012	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	455	Certificate (Reassembled), Server Key Exchange, Certificate Request, Server Hello Done
12	0.029479	172.31.0.4	28951	172.31.0.3	5001	DTLSv1.2	1514	Certificate (Fragment)
13	0.029505	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	1514	Certificate (Fragment)
14	0.032030	172.31.0.4	28951	172.31.0.3	5001	DTLSv1.2	773	Certificate (Reassembled), Client Key Exchange, Certificate Verify, Change Cipher Spec, Encrypted Handshake Message
15	0.035015	172.31.0.3		172.31.0.4		IPv4	1514	Fragmented IP protocol (proto=UDP 17, off=0, ID=cid3) [Reassembled in #16]
16	0.035039	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	58	New Session Ticket, Change Cipher Spec, Encrypted Handshake Message
17	0.035107	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	93	Encrypted Handshake Message
18	5.006550	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	Who has 172.31.0.4 Tell 172.31.0.3
19	5.006926	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3 Tell 172.31.0.4
20	5.006960	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0
21	5.007350	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	172.31.0.4 is at 00:16:3e:3d:17:94
22	21.905318	172.31.0.4	28951	172.31.0.3	5001	DTLSv1.2	104	Application Data → Hi bob1
23	29.764940	172.31.0.3	5001	172.31.0.4	28951	DTLSv1.2	88	Application Data → Hi alice1
24	41.614778	Xensource_3d:17:94		Xensource_d2:a2:f0		ARP	42	Who has 172.31.0.3 Tell 172.31.0.4
25	41.614804	Xensource_d2:a2:f0		Xensource_3d:17:94		ARP	42	172.31.0.3 is at 00:16:3e:d2:a2:f0

```
> Frame 23: 88 bytes on wire (704 bits), 88 bytes captured (704 bits)
> Ethernet II, Src: Xensource_d2:a2:f0 (00:16:3e:d2:a2:f0), Dst: Xensource_3d:17:94 (00:16:3e:3d:17:94)
> Internet Protocol Version 4, Src: 172.31.0.3, Dst: 172.31.0.4
> User Datagram Protocol, Src Port: 28951, Dst Port: 28951
└ DTLSv1.2 Record Layer: Application Data Protocol: Application Data
    Content Type: Application Data (23)
    Version: DTLS 1.2 (0xfeed)
    Epoch: 1
    Sequence Number: 2
    Length: 33
    Encrypted Application Data: d7b29d055b73e8e1e9b6afa017d5357e0b4b94e99dfde35da05ab53fd1c461018d0
```

- In the above screenshot, we can observe that the last chat message sent by bob1 (172.31.0.3) to trudy1 (172.31.0.4) is 'Hi alice1'.
- We can see that bob1 has not received the last message 'How are you?' sent by alice1.

## Tcpdump trace captured at trudy1:



- From the above screenshot, we can observe that the first 2 chat messages 'Hi bob1' and 'Hi alice1' have been successfully exchanged between alice1 and bob1.
- However, the last chat message 'How are you?' sent from trudy1 to bob1 got lost due to injected packet loss.

### Answer:

- DTLS v1.2 protocol doesn't provide reliability. Hence we can see that few chat messages got lost during communication.
- To avoid this packet loss, we must implement some reliable data transfer mechanism to exchange the DTLS v1.2 messages.

## Credit Statement

### Task 1 contributions:

	Hrishikesh Hemke (CS23MTECH14003)	Mayuresh Dindorkar (CS23MTECH14007)	Shrenik Ganguli (CS23MTECH14014)
Software Installation	OpenSSL version 3.0.2		
Coding	Generated keys, CSR, digests and Certificates (CRT) for Root Certification Authority (Root CA) and verified them.	Generated keys, CSR, digests and Certificates (CRT) for Intermediate Certification Authority (Int CA) and verified them from Root CA.	Generated keys, CSR, digests and Certificates (CRT) for Alice1 and Bob1 and verified them from Intermediate CA.
Report writing	Written report for Root CA.	Written report for Intermediate CA.	Written report for Alice1 and Bob1.
Bug fixes	<p><i>Bug:</i> The trust chain was not getting formed in the alice1 and bob1's certificate.</p> <p><i>Resolution:</i> Use the configuration and extension file.</p>		

### Task 2 contributions:

	Hrishikesh Hemke (CS23MTECH14003)	Mayuresh Dindorkar (CS23MTECH14007)	Shrenik Ganguli (CS23MTECH14014)
Coding	Run the commands to capture the tcpdump pcap traces at both alice1 and bob1's end.  Also, injected the packet loss in the secure chat application using 'tc-netem' command and captured the corresponding pcap files.	Implemented the DTLS v1.2 connection between alice1 and bob1 using openssl in C programming language.  Implemented the certificate exchange and verification logic during DTLS v1.2 handshake.  Also implemented the cipher suites negotiation logic.	Implemented the exchange of control messages between alice1 and bob1 using UDP sockets.  Studied the difference between DTLS 1.2 and TLS 1.2 handshakes.
Testing	Tested the cipher suites negotiation functionality.	Tested the correctness of control message exchange between alice1 and bob1.	Tested the certificate exchange and verification logic during DTLS v1.2 handshake.
Report writing	Written the report for the above work.	Written the report for the above work.	Written the report for the above work.

<b>Bug fixes</b>	<i>Bug:</i> ‘key too small’ error for 1024-bit RSA key of alice1. <i>Resolution:</i> Used a 2048-bit RSA key for alice1.
<b>Additional features</b>	<ol style="list-style-type: none"> <li>1. The server (bob1) keeps on listening for new client connection requests when the chat with the current client is closed.</li> <li>2. Trimmed the preceding as well as trailing white spaces present in the chat messages.</li> </ol>

### Task 3 contributions:

	<b>Hrishikesh Hemke (CS23MTECH14003)</b>	<b>Mayuresh Dindorkar (CS23MTECH14007)</b>	<b>Shrenik Ganguli (CS23MTECH14014)</b>
<b>Coding</b>	<p>Modified the control message flow to incorporate ‘START_SSL_NOT_SUPPORTED’ message.</p> <p>Implemented the logic in the code to divert the execution from secure to unsecure chat.</p>	<p>Implemented the unsecure chat using two UDP sockets:</p> <ol style="list-style-type: none"> <li>1. ‘trudy_alice_socket’ between trudy1 and alice1.</li> <li>2. ‘Trudy_bob_socket’ between trudy1 and bob1.</li> </ol>	<p>Designed the architecture of all modules involved in code including trudy1.</p> <p>Run the commands to capture the tcpdump pcap traces at alice1, bob1 and trudy1 containers.</p>
<b>Report writing</b>	Written the report for the above work.	Written the report for the above work.	Written the report for the above work.
<b>Additional features</b>	<ol style="list-style-type: none"> <li>1. The server (bob1) keeps on listening for new client connection requests when the chat with the current client is closed.</li> <li>2. Trimmed the preceding as well as trailing white spaces present in the chat messages.</li> </ol>		

**Task 4 contributions:**

	<b>Hrishikesh Hemke (CS23MTECH14003)</b>	<b>Mayuresh Dindorkar (CS23MTECH14007)</b>	<b>Shrenik Ganguli (CS23MTECH14014)</b>
<b>Coding</b>	<p>Run the commands to capture the tcpdump pcap traces at alice1, bob1 and trudy1 containers.</p> <p>Injected the packet loss in the secure chat application using 'tc-netem' command and captured the corresponding pcap files.</p>	<p>Implemented the two DTLS v1.2 pipes:</p> <ol style="list-style-type: none"> <li>1. Between trudy1 and alice1</li> <li>2. Between trudy1 and bob1</li> </ol> <p>Also implemented the logic for transfer of fake certificates from trudy1 to alice1 as well as bob1.</p>	<p>Generated the fake certificates (CRT), CSR, digests and keys for alice1 and bob1.</p> <p>Verified these fake certificates using intermediate CA.</p>
<b>Report writing</b>	Written the report for the above work.	Written the report for the above work.	Written the report for the above work.

### **ANTI-PLAGIARISM STATEMENT**

*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: Hrishikesh Hemke , Dindorkar Mayuresh Rajesh , Shrenik Ganguli

Date: 07/04/2024

Signature: HH, MRD, SG