Secure chat using Openssl and MITM attacks on it

Group-2

Chat Application Messages:

- **Chat message sent by Alice to Bob**

chat_hello
chat_STARTTLS
chat_close

- **Chat message sent by Bob to Alice**

chat_reply
chat_STARTTLS_ACK
chat_STARTTLS_NOTSUPPORTED

Important Steps:

Task-1:

Use the OpenSSL commands to create a root CA certificate (V3 X.509 certificate, self-signed using 512-bit ECC Private Key of the root), a certificate of Alice (issued i.e., signed by the root CA) and a certificate of Bob (issued by the root CA).

Step-1: Generate a 512-bit ECC private key for the root.

```
ns@ns02:~/Task1$ openssl genpkey -algorithm EC -out root_key.pem -pkeyopt ec_paramgen_curve:brainpoolP512t1 -aes256
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
ns@ns02:~/Task1$
```

Note: brainpoolP512t1 is the name of the standard elliptical curve which produces 512 bit private keys.

Step-2: Separate the public key of the root and store it.

```
ns@ns02:~/Task1$ openssl pkey -in root_key.pem -pubout -out root_public.pem
Enter pass phrase for root_key.pem:
ns@ns02:~/Task1$
```

Step-3: Create a self signed CA certificate for the root.

```
ns@ns02:~/Task1$ openssl req -x509 -key root_key.pem -out root.crt -days 3650
Enter pass phrase for root_key.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IITH
Organizational Unit Name (eg, section) []:NS
Common Name (e.g. server FQDN or YOUR name) []:www.group2.com
Email Address []:cs21mtech12009@iith.ac.in
ns@ns02:~/Task1$
```

The resulting self-signed root CA certificate is as follows:

```
ns@ns02:~/Task1$ openssl x509 -in root.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            7e:80:00:fb:71:ea:15:1c:c2:6d:79:7c:81:a5:13:18:2b:23:df:07
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = NS, CN = www.group2.com, emailAddress = cs21mtech12009@iith.ac.in
        Validity
            Not Before: Mar 31 06:31:05 2022 GMT
            Not After : Mar 28 06:31:05 2032 GMT
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = NS, CN = www.group2.com, emailAddress = cs21mtech12009@iith.ac.in
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (512 bit)
                pub:
                    04:7e:63:ca:03:88:fb:bf:04:2e:18:8e:04:71:63:
                    2d:e3:e5:f4:5e:8f:96:7b:67:aa:ed:ef:d4:61:74:
                    1f:4a:b5:70:f4:7f:aa:61:9a:b5:31:15:13:02:52:
                    fc:e7:3b:bf:51:90:5d:f1:f7:63:50:cd:a9:5e:76:
                    fb:4b:20:05:23:4f:0a:e0:6f:d2:c5:67:7d:e9:6f:
                    6f:99:ab:5e:fd:c6:89:ac:71:5f:86:db:aa:38:82:
                    97:bf:e9:33:1a:96:70:5b:db:4f:3c:35:ee:10:9d:
                    ea:aa:eb:1c:35:3f:43:94:5a:37:99:07:4f:b9:04:
                    7f:55:70:63:b1:89:94:4d:28
                ASN1 OID: brainpoolP512t1
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                0F:CD:30:D3:E5:AD:20:34:A9:C5:DC:3F:AC:88:5B:56:75:BE:8C:43
            X509v3 Authority Key Identifier:
                keyid:0F:CD:30:D3:E5:AD:20:34:A9:C5:DC:3F:AC:88:5B:56:75:BE:8C:43

            X509v3 Basic Constraints: critical
                CA:TRUE
    Signature Algorithm: ecdsa-with-SHA256
         30:81:85:02:40:1f:f0:08:b3:b7:10:f1:09:d8:a3:2b:38:cd:
         a7:6a:77:bd:d9:09:91:b1:78:7d:c5:fd:3a:54:b0:6f:79:55:
         b2:eb:e4:65:4e:16:bb:79:0a:3d:b4:c1:11:47:b7:d2:9c:1e:
         f5:ea:43:71:ad:5a:df:1f:85:8f:24:6c:7b:1f:29:02:41:00:
         96:2f:4b:cf:8e:ac:48:2b:f9:58:db:d5:d8:b2:c5:3e:f3:30:
         d9:82:de:c2:8b:d3:12:28:f5:b6:3b:c5:21:fd:61:99:e7:e8:
         00:48:d0:c8:8d:29:16:59:07:bc:d3:44:63:1a:93:91:82:47:
         b0:fb:53:39:d2:19:97:6c:47:c4
ns@ns02:~/Task1$
```

Step-4: Generate a RSA private key for Alice and create a CSR to be signed by the root CA.

```
ns@ns02:~/Task1/RSA$ openssl genrsa -out alice_key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................+++++
.....................................+++++
e is 65537 (0x010001)
ns@ns02:~/Task1/RSA$ openssl rsa -in alice_key.pem -des3 -out alice_private.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
ns@ns02:~/Task1/RSA$ openssl rsa -in alice_private.pem -pubout -out alice_public.pem
Enter pass phrase for alice_private.pem:
writing RSA key
ns@ns02:~/Task1/RSA$ openssl req -new -key alice_private.pem -out alice.csr
Enter pass phrase for alice_private.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:IITH
Organizational Unit Name (eg, section) []:A block
Common Name (e.g. server FQDN or YOUR name) []:www.alice.com
Email Address []:cs21mtech12009@iith.ac.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:alice123
An optional company name []:A
ns@ns02:~/Task1/RSA$
```

Step-5: Get Alice's CSR signed by the root CA and verify Alice's certificate.

```
ns@ns02:~/Task1/RSA$ openssl x509 -req -days 360 -extfile v3.ext -in alice.csr -CA root.crt -CAkey root_key.pem -CAcreateserial -out ali
ce.crt
Signature ok
subject=C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = A block, CN = www.alice.com, emailAddress = cs21mtech12009@iith.ac.in
Getting CA Private Key
Enter pass phrase for root_key.pem:
ns@ns02:~/Task1/RSA$ openssl verify -CAfile root.crt alice.crt
alice.crt: OK
ns@ns02:~/Task1/RSA$
```

Step-6: Repeat this process for Bob

```
ns@ns02:~/Task1/RSA$ openssl genrsa -out bob_key.pem 2048
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................................................+++++
.....................+++++
e is 65537 (0x010001)
ns@ns02:~/Task1/RSA$ openssl rsa -in bob_key.pem -des3 -out bob_private.pem
writing RSA key
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
ns@ns02:~/Task1/RSA$ openssl rsa -in bob_private.pem -pubout -out bob_public.pem
Enter pass phrase for bob_private.pem:
writing RSA key
ns@ns02:~/Task1/RSA$ openssl req -new -key bob_private.pem -out bob.csr
Enter pass phrase for bob_private.pem:
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:Telangana
Locality Name (eg, city) []:Hyderabad
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Acad
Organizational Unit Name (eg, section) []:C Block
Common Name (e.g. server FQDN or YOUR name) []:www.bob.com
Email Address []:cs21mtech12009@iith.ac.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:bob123
An optional company name []:C
ns@ns02:~/Task1/RSA$
```

```
ns@ns02:~/Task1/RSA$ openssl x509 -req -days 360 -extfile v3.ext -in bob.csr -CA root.crt -CAkey root_key.pem -CAcreateserial -out bob.crt
Signature ok
subject=C = IN, ST = Telangana, L = Hyderabad, O = Acad, OU = C Block, CN = www.bob.com, emailAddress = cs21mtech12009@iith.ac.in
Getting CA Private Key
Enter pass phrase for root_key.pem:
```

```
ns@ns02:~/Task1/RSA$ openssl verify -CAfile root.crt bob.crt
bob.crt: OK
```

Resulting Bob.crt looks like this:

```
ns@ns02:~/Task1/RSA$ openssl x509 -in bob.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            04:02:df:f9:79:32:c1:f5:4f:dc:d5:14:ad:61:f0:09:da:7c:d8:45
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = NS, CN = www.group2.com, emailAddress = cs21mtech12009@iith.ac.in
        Validity
            Not Before: Apr  5 02:04:18 2022 GMT
            Not After : Mar 31 02:04:18 2023 GMT
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = Acad, OU = C Block, CN = www.bob.com, emailAddress = cs21mtech12009@iith.ac.in
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:cf:72:c3:54:13:55:7a:60:a2:7a:29:71:7d:f6:
                    9d:b0:2c:dd:43:03:c3:c3:e2:4b:2c:08:fe:1d:20:
                    b6:24:21:0d:ef:0c:0a:c2:f8:4f:3c:27:f1:b1:34:
                    d8:7d:8f:ac:71:a2:f1:ec:c0:75:fb:79:e1:c6:f6:
                    00:ae:fd:d3:22:fb:86:b6:e2:0b:7f:b7:34:c3:0c:
                    f0:da:70:0c:d3:c5:18:d5:f7:4d:3d:a8:07:bb:90:
                    6e:d7:9f:fb:70:1a:5e:50:7b:ff:e8:27:b2:64:50:
                    b4:24:ca:19:f0:07:d2:90:dc:dd:02:a0:e1:fa:c4:
                    56:9e:30:32:70:47:25:2a:38:b6:fe:1c:6c:32:b3:
                    b4:1b:e1:b4:f1:06:d1:1a:cb:46:96:96:75:6b:d8:
                    16:76:a6:0d:ca:75:06:1c:f7:1b:8e:4c:4b:d4:69:
                    fb:19:6d:9b:bc:f7:f7:c4:1a:c0:b6:4e:31:ef:57:
                    9b:07:ec:86:ac:5a:7d:64:cf:57:a5:60:78:f3:65:
                    34:49:bc:63:2a:b4:03:06:07:dc:23:dd:2a:02:b8:
                    20:63:c1:32:cf:59:e2:fc:f1:19:c2:08:d0:d0:a1:
                    6f:0c:30:75:2c:0b:33:6f:9b:09:ed:9e:f1:6f:03:
                    a5:9c:19:2d:d0:51:e8:62:72:d6:53:e7:0d:8d:77:
                    ec:e3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:0F:CD:30:D3:E5:AD:20:34:A9:C5:DC:3F:AC:88:5B:56:75:BE:8C:43

            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data Encipherment
    Signature Algorithm: ecdsa-with-SHA256
         30:81:84:02:40:4c:bd:a4:2a:d9:46:24:8b:f0:13:07:3b:d1:
         a4:ec:08:34:36:4d:40:20:bf:0b:b3:a0:3a:32:52:b1:1f:5d:
         99:c1:8a:b1:51:0e:9a:66:12:49:c4:42:5f:7a:fc:b8:ab:25:
         0e:8b:53:24:30:a2:65:38:ae:01:0f:d7:07:08:23:02:40:13:
         08:6f:41:71:9a:31:7c:86:40:fc:fb:41:e8:48:5c:6f:bd:25:
```

Resulting Alice.crt looks like this:

```
ns@ns02:~/Task1$ openssl x509 -in alice.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            04:02:df:f9:79:32:c1:f5:4f:dc:d5:14:ad:61:f0:09:da:7c:d8:43
        Signature Algorithm: ecdsa-with-SHA256
        Issuer: C = IN, ST = Telangana, L = Hyderabad, O = IITH, OU = NS, CN = w
ww.group2.com, emailAddress = cs21mtech12009@iith.ac.in
        Validity
            Not Before: Mar 31 07:21:49 2022 GMT
            Not After : Mar 26 07:21:49 2023 GMT
        Subject: C = IN, ST = Telangana, L = Hyderabad, O = Alice Pvt Ltd, OU =
Canteen, CN = www.abc.com, emailAddress = cs21mtech12009@iith.ac.in
        Subject Public Key Info:
            Public Key Algorithm: id-ecPublicKey
                Public-Key: (512 bit)
                pub:
                    04:81:31:96:0c:88:58:69:f5:08:2f:eb:2b:90:a7:
                    bd:a9:db:e7:c0:49:cb:5d:1d:ba:73:48:98:85:a8:
                    de:e1:67:bd:07:3c:fe:a5:00:15:a0:c7:7f:0a:f1:
                    0e:f3:bc:37:10:2b:a4:46:d0:1d:7b:58:fb:8e:31:
                    cd:ec:46:65:6e:02:f0:7e:38:39:c2:cb:cf:64:24:
                    08:c5:1c:05:14:cc:55:93:2c:d3:df:df:62:07:5d:
                    aa:e7:68:ce:57:6d:61:84:f6:19:87:e0:2a:64:81:
                    3f:8c:1a:fd:21:a7:02:7a:b7:b3:39:81:45:1f:f9:
                    71:0d:35:d3:54:fd:a0:f9:0c
                ASN1 OID: brainpoolP512t1
        X509v3 extensions:
            X509v3 Authority Key Identifier:
                keyid:0F:CD:30:D3:E5:AD:20:34:A9:C5:DC:3F:AC:88:5B:56:75:BE:8C:4
3

            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment, Data Encip
herment
    Signature Algorithm: ecdsa-with-SHA256
         30:81:84:02:40:62:22:62:1f:9b:03:5f:ee:5a:c4:9f:6b:bc:
         51:64:8b:a3:71:c6:30:18:11:af:63:e6:8e:d1:9c:92:3c:dc:
         94:27:2e:11:b1:e6:be:ba:09:5f:d1:f5:46:27:71:30:b9:35:
         ea:e3:14:d6:90:d1:af:90:bb:3a:a6:46:59:fb:5e:02:40:1f:
         37:d2:1c:30:7f:f8:a8:19:2c:ca:1e:52:51:32:9a:16:62:05:
         35:7d:3a:f2:00:b8:d6:34:6d:15:7c:03:08:92:b9:fc:a8:2a:
         14:6a:38:f8:c6:46:e2:60:1c:90:0a:89:0b:9b:4a:18:31:1d:
         27:77:d7:b6:c1:d6:34:c0:9a
```

Task-2:

Write a peer-to-peer application (`secure_chat_app`) for chatting which uses TLS 1.3 and TCP as the underlying protocols for secure and reliable communication.

1. **chat_hello and chat_reply**

- Alice sends chat_hello to initiate the chat application, in reply Bob sends a chat_reply message to Alice.
- These messages are sent in plain text and can be seen in the following screenshots.

```
Frame 4: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
Ethernet II, Src: Xensourc_d0:af:c8 (00:16:3e:d0:af:c8), Dst: Xensourc_89:0d:45 (00:16:3e:89:0d:45)
Internet Protocol Version 4, Src: 172.31.0.2, Dst: 172.31.0.3
Transmission Control Protocol, Src Port: 36262, Dst Port: 9000, Seq: 1, Ack: 1, Len: 10
Data (10 bytes)
   Data: 636861745f68656c6c6f
   [Length: 10]
```

```
000  00 16 3e 89 0d 45 00 16  3e d0 af c8 08 00 45 00   ··>··E·· >·····E·
010  00 3e 6c 58 40 00 40 06  76 1e ac 1f 00 02 ac 1f   ·>lX@·@· v·······
020  00 03 8d a6 23 28 d9 5b  34 f6 ec 0d fc 5c 80 18   ····#(·[ 4····\··
030  01 f6 58 74 00 00 01 01  08 0a ce cf 50 4f a2 80   ··Xt···· ····PO··
040  ad 99 63 68 61 74 5f 68  65 6c 6c 6f               ··chat_h ello
```

chat_hello in plain text in pcap trace

```
> Frame 6: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
> Ethernet II, Src: Xensourc_89:0d:45 (00:16:3e:89:0d:45), Dst: Xensourc_d0:af:c8 (00:16:3e:d0:af:c8)
> Internet Protocol Version 4, Src: 172.31.0.3, Dst: 172.31.0.2
> Transmission Control Protocol, Src Port: 9000, Dst Port: 36262, Seq: 1, Ack: 11, Len: 10
v Data (10 bytes)
     Data: 636861745f7265706c79
     [Length: 10]
```

```
0000  00 16 3e d0 af c8 00 16  3e 89 0d 45 08 00 45 00   ··>····· >··E··E·
0010  00 3e 36 b7 40 00 40 06  ab bf ac 1f 00 03 ac 1f   ·>6·@·@· ········
0020  00 02 23 28 8d a6 ec 0d  fc 5c d9 5b 35 00 80 18   ··#(···· ·\·[5···
0030  01 fd 58 74 00 00 01 01  08 0a a2 80 ad 99 ce cf   ··Xt···· ········
0040  50 4f 63 68 61 74 5f 72  65 70 6c 79               POchat_r eply
```

chat_reply in plain text in pcap trace

2. **The Root CA certificate is stored in the Trust store of Alice and Bob**

- The CA certificate (root.crt) is added to the Trust Store of Alice and Bob as shown below in the screenshot.

```
root@bob1:~# cp root.crt /usr/local/share/ca-certificates/
root@bob1:~# sudo update-ca-certificates
Updating certificates in /etc/ssl/certs...
1 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
root@bob1:~# _
```

3. chat_STARTTLS and chat_STARTTLS_ack

- Alice sends chat_STARTTLS to establish a TLSv1.3 pipe, in reply Bob sends a chat_STARTTLS_ack message to Alice after successful establishment of TLS.
- The program loads private keys and certificates of Alice and Bob and exchanges them.
- After that, both the parties communicate in secure fashion, messages are encrypted and can be seen in the following screenshots.

18	45.119940	172.31.0.3	172.31.0.2	9000	36262	TLSv1.3	2327 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data, Application
19	45.119978	172.31.0.2	172.31.0.3	36262	9000	TCP	66 36262 → 9000 [ACK] Seq=207 Ack=2289 Win=63872 [TCP CHECKSUM INCORRECT] Len=0 TSval=3469738127 TSecr=
20	45.127777	172.31.0.2	172.31.0.3	36262	9000	TLSv1.3	2133 Application Data, Application Data, Application Data
21	45.127801	172.31.0.3	172.31.0.2	9000	36262	TCP	66 9000 → 36262 [ACK] Seq=2289 Ack=2274 Win=63872 [TCP CHECKSUM INCORRECT] Len=0 TSval=2726387169 TSecr
22	45.129802	172.31.0.3	172.31.0.2	9000	36262	TLSv1.3	1217 Application Data
23	45.129819	172.31.0.2	172.31.0.3	36262	9000	TCP	66 36262 → 9000 [ACK] Seq=2274 Ack=3440 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=3469738137 TSecr
24	45.129918	172.31.0.3	172.31.0.2	9000	36262	TLSv1.3	1217 Application Data
25	45.129925	172.31.0.2	172.31.0.3	36262	9000	TCP	66 36262 → 9000 [ACK] Seq=2274 Ack=4591 Win=63872 [TCP CHECKSUM INCORRECT] Len=0 TSval=3469738137 TSecr
26	50.269356	Xensourc_89:0d:45	Xensourc_d0:af:c8			ARP	42 Who has 172.31.0.2? Tell 172.31.0.3
27	50.271335	Xensourc_d0:af:c8	Xensourc_89:0d:45			ARP	42 Who has 172.31.0.3? Tell 172.31.0.2
28	50.271357	Xensourc_89:0d:45	Xensourc_d0:af:c8			ARP	42 172.31.0.3 is at 00:16:3e:89:0d:45
29	50.271369	Xensourc_d0:af:c8	Xensourc_89:0d:45			ARP	42 172.31.0.2 is at 00:16:3e:d0:af:c8
30	132.272…	172.31.0.2	172.31.0.3	36262	9000	TLSv1.3	97 Application Data
31	132.272…	172.31.0.3	172.31.0.2	9000	36262	TCP	66 9000 → 36262 [ACK] Seq=4591 Ack=2305 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=2726474314 TSecr
32	136.396…	172.31.0.3	172.31.0.2	9000	36262	TLSv1.3	97 Application Data
33	136.396…	172.31.0.2	172.31.0.3	36262	9000	TCP	66 36262 → 9000 [ACK] Seq=2305 Ack=4622 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=3469829404 TSecr
34	137.309…	Xensourc_89:0d:45	Xensourc_d0:af:c8			ARP	42 Who has 172.31.0.2? Tell 172.31.0.3
35	137.309…	Xensourc_d0:af:c8	Xensourc_89:0d:45			ARP	42 172.31.0.2 is at 00:16:3e:d0:af:c8
36	148.710…	172.31.0.2	172.31.0.3	36262	9000	TLSv1.3	103 Application Data
37	148.710…	172.31.0.3	172.31.0.2	9000	36262	TCP	66 9000 → 36262 [ACK] Seq=4622 Ack=2342 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=2726490751 TSecr
38	152.641…	172.31.0.3	172.31.0.2	9000	36262	TLSv1.3	97 Application Data
39	152.641…	172.31.0.2	172.31.0.3	36262	9000	TCP	66 36262 → 9000 [ACK] Seq=2342 Ack=4653 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=3469845648 TSecr
40	163.681…	172.31.0.2	172.31.0.3	36262	9000	TLSv1.3	98 Application Data
41	163.681…	172.31.0.3	172.31.0.2	9000	36262	TCP	66 9000 → 36262 [ACK] Seq=4653 Ack=2374 Win=64128 [TCP CHECKSUM INCORRECT] Len=0 TSval=2726505722 TSecr

After Server Hello, change cipher spec message, all the chat data is sent in encrypted mode

4. Secure chat between Alice and Bob

```
root@alice1:~# python3 secure_chat_app.py -c bob1
Message from Bob:  chat_reply
Message from Bob:  chat_STARTTLS_ACK
Enter PEM pass phrase:
Bob Certificate Verified Succesfully....
TLSv1.3 Handshake Completed....
TLSv1.3 established Succesfully....
#### Secure Chat ######
Enter message to send it to Bob: Hello Bob
Message from Bob:  Hey Alice
Enter message to send it to Bob: How are you? :)
Message from Bob:  I am good
Enter message to send it to Bob: chat_close
Closing TLSv1.3 and TCP Connection
root@alice1:~#
```

Alice

```
root@bob1:~# python3 secure_chat_app.py -s
Bob listening on IP: 172.31.0.3
 Waiting for Alice to connect......
Message from Alice:  chat_hello
Message from Alice:  chat_STARTTLS
Enter PEM pass phrase:
Alice Certificate Verified Succesfully....
TLSv1.3 Handshake Completed....
TLSv1.3 established Succesfully....
#### Secure Chat ######
Message from Alice:  Hello Bob
Enter message to send it to Alice: Hey Alice
Message from Alice:  How are you? :)
Enter message to send it to Alice: I am good
Closing TLSv1.3 and TCP Connection
root@bob1:~#
```

Bob

5. **chat_close**

- As soon as Alice wants to close the chat application, it sends chat_close to Bob.
- Both of them closes the TLSv1.3 and TCP connection

6. **task2.pcap is captured at bob1**

Task 3:

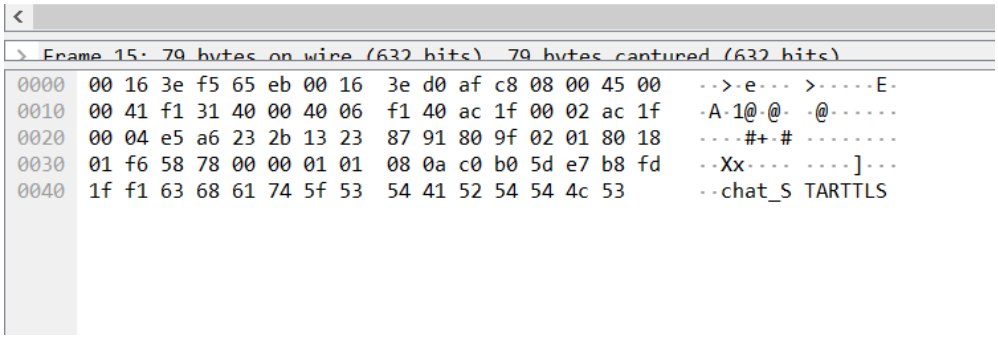
Downgrade attack by Trudy by blocking the chat_STARTTLS message from Alice (Bob) to Bob (Alice).

1. Trudy poisons the DNS Server of Alice, so IP of Bob **(172.31.0.3)** is replaced by IP of Trudy **(172.31.0.4)**. So, when Alice tries to connect to Bob, it actually gets connected to Trudy.

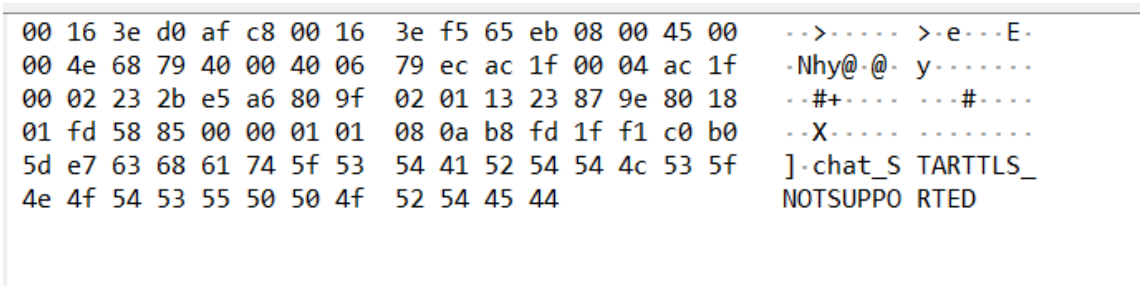
2. For this task, Trudy has created 2 sockets → fakebob socket and fakealice socket:

    - **fakebob_sock** → True Alice gets connected to Fake Bob, which helps block the chat_STARTTLS message
    - **fakealice_sock** → The messages received by fakebob_sock or True Bob is forwarded to True bob or True Alice via this socket
    - **alice_sock** → After True alice connects to fakebob_sock we store the connection to this

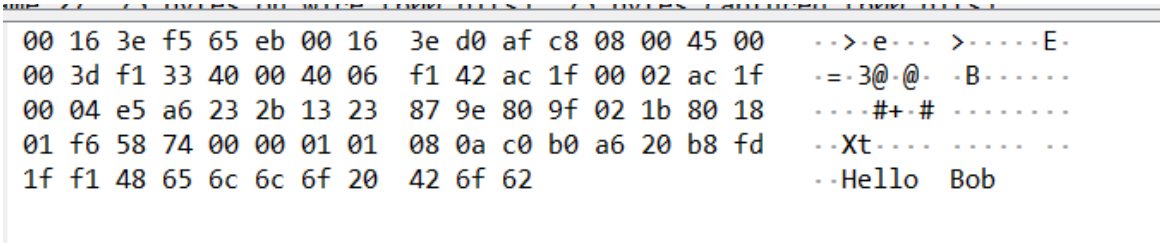
3. As soon as fakebob_sock receives a chat_STARTTLS message, it sends chat_STARTTLS_NOTSUPPORTED to True alice via alice_sock launching a successful TLS downgrade attack.



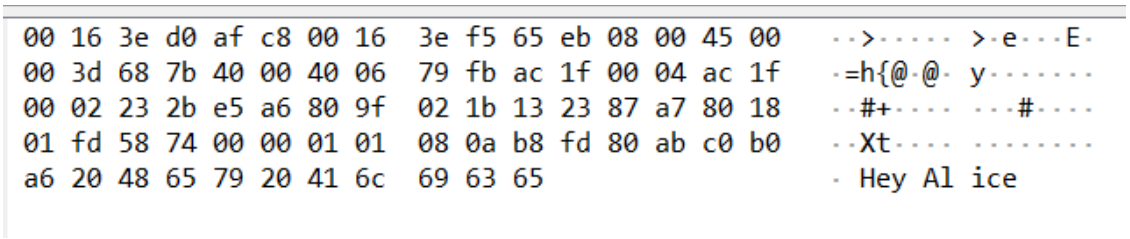
Alice sends chat_STARTTLS, which is blocked by Trudy



Trudy sends chat_STARTTLS_NOTSUPPORTED upon receiving chat_STARTTLS from Alice



Trudy forwards the message “Hello Bob” from Alice to Bob



Trudy forwards the message “Hey Alice” from Bob to Alice

4. So, Alice will start unsecure chat application with True Bob, all the normal traffic is forwarded in plaintext to and fro Alice and Bob via Trudy using fakebob_sock and fakealice_sock and can be seen in the above screenshots.

```
ns@ns02:~$ lxc exec alice1 bash
root@alice1:~# python3 secure_chat_app.py -c bob1
IP of Bob is:  172.31.0.4
Message from Bob:  chat_reply
chat_STARTTLS sent to Bob
TLS not supported by Bob....
#### Unsecure Chat ######
Enter message to send it to Bob: Hello Bob
Message from Bob:  Hey Alice
Enter message to send it to Bob: chat_close
Closing TCP Connection
root@alice1:~#
```

Alice

```
ns@ns02:~$ lxc exec bob1 bash
root@bob1:~# python3 secure_chat_app.py -s
Bob listening on IP: 172.31.0.3
 Waiting for Alice to connect......
Message from Alice:  chat_hello
Message from Alice:  Hello Bob
Enter message to send it to Alice: Hey Alice
Message from Alice:  chat_close
TCP Connection closed!
root@bob1:~#
```

Bob

```
ns@ns02:~$ lxc exec trudy1 bash
root@trudy1:~# python3 secure_chat_interceptor.py -d alice1 bob1
Fake Bob listening ...
 Waiting for Alice to connect......
True Alice Connected to Fake Bob with IP:  172.31.0.4
Fake Alice Connected to True Bob with IP:  172.31.0.3
Message from True Alice:  chat_hello
Forwarding to Bob as it as ....
Message from True Bob:  chat_reply
Forwarding to Alice as it as ....  chat_reply
Message from True Alice:  chat_STARTTLS
Block, send STARTTLS_NOTSUPPORTED to True Alice ....  chat_STARTTLS
Successfully launched TLS Downgrade Attack....
Message from True Alice:  Hello Bob
Forwarding to Bob as it as ....
Message from True Bob:  Hey Alice
Forwarding to Alice as it as ....  Hey Alice
Message from True Alice:  chat_close
Forwarding to Bob as it as ....
root@trudy1:~#
```

Trudy launching TLS downgrade attack

5. **The task3.pcap file is captured at Trudy1**

Task 4:

Active MITM attack by Trudy to tamper the chat communication between Alice and Bob. For this task also, you can assume that Trudy poisoned the /etc/hosts file of Alice (Bob) and replaced the IP address of Bob (Alice) with that of her.

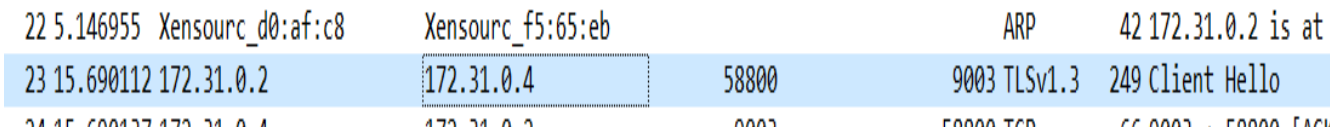
1. Trudy poisons the DNS Server of Alice, so IP of Bob **(172.31.0.3)** is replaced by IP of Trudy **(172.31.0.4)**. So, when Alice tries to connect to Bob, it actually gets connected to Trudy.

2. Similar to Task1, we generate fake alice and bob certificates and the root.crt (CA certificate) is added to the Trust Store of Trudy.

3. For this task also, Trudy has created 2 sockets → fakebob socket and fakealice socket:

    - **fakebob_sock** → True Alice gets connected to Fake Bob, which helps block the chat_STARTTLS message
    - **fakealice_sock** → The messages received by fakebob_sock or True Bob is forwarded to True bob or True Alice via this socket
    - **alice_sock** → After True alice connects to fakebob_sock we store the connection to this

4. As soon as Trudy receives (fakebob_sock) chat_STARTTLS from True Alice, it sends Fake Bbob certificate to True Alice during TLS certificate exchange. Similarly, Trudy sends a Fake Alice certificate to True Bob. So, Trudy establishes 2 TLS pipes, one between True Alice and Fake Bob and another between Fake Alice and True Bob. The 2 TLS pipes can be seen in task4.pcap trace, and the screenshot below:



TLSv1.3 from True Alice (172.31.0.2) to Fake Bob

42 31.002782 Xensourc_89:0d:45	Xensourc_f5:65:eb			ARP	42	172.31.0.3 is
43 34.494251 172.31.0.4	172.31.0.3	53712	9003	TLSv1.3	249	Client Hello
44 34.494351 172.31.0.3	172.31.0.4	9003	53712	TCP	66	9003 → 53712 [

TLSv1.3 from Fake Alice to True Bob(172.31.0.3)

5. So, any message that Tru Alice sends to True Bob or vice versa is passed through Trudy and she can launch MITM by tampering the messages shown below:

```
root@alice1:~# python3 secure_chat_app.py -c bob1
IP of Bob is:  172.31.0.4
Message from Bob:  chat_reply
chat_STARTTLS sent to Bob
Message from Bob:  chat_STARTTLS_ACK
Enter PEM pass phrase:
Bob Certificate Verified Succesfully....
TLSv1.3 Handshake Completed....
TLSv1.3 established Succesfully....
#### Secure Chat ######
Enter message to send it to Bob: Hello Bob, this is Alice!
Message from Bob:  This message from Bob is distorted by Trudy--->0
Enter message to send it to Bob: Are you reciving my messages?
Message from Bob:  Yes
Enter message to send it to Bob: chat_close
Closing TLSv1.3 and TCP Connection
root@alice1:~#
```

Alice

```
root@bob1:~# root@bob1:~# python3 secure_chat_app.py -s
Bob listening on IP: 172.31.0.3
 Waiting for Alice to connect......
Message from Alice:  chat_hello
Message from Alice:  chat_STARTTLS
Enter PEM pass phrase:
Alice Certificate Verified Succesfully....
TLSv1.3 Handshake Completed....
TLSv1.3 established Succesfully....
#### Secure Chat ######
Message from Alice:  This message from Alice is distorted by Trudy--->0
Enter message to send it to Alice: Hey Alice!
Message from Alice:  Are you reciving my messages?
Enter message to send it to Alice: Yes
Closing TLSv1.3 and TCP Connection
root@bob1:~#
```

Bob

```
root@trudy1:~# root@trudy1:~# python3 secure_chat_interceptor.py -m alice1 bob1
Fake Bob listening ...
 Waiting for Alice to connect......
True Alice Connected to Fake Bob with IP:  172.31.0.4
Fake Alice Connected to True Bob with IP:  172.31.0.3
Message from True Alice:  chat_hello
Sent chat_reply....
Message from True Alice:  chat_STARTTLS
Enter PEM pass phrase:
Alice Certificate Verified Succesfully....
TLSv1.3 Handshake Completed With Real Alice....
TLSv1.3 established Succesfully With Real Alice....
Message from Bob:  chat_reply
Message from Bob:  chat_STARTTLS_ACK
Enter PEM pass phrase:
Bob Certificate Verified Succesfully....
TLSv1.3 Handshake Completed With Real Bob....
TLSv1.3 established Succesfully With Real Bob....
#### Secure Chat ######
Original Message from Alice:  Hello Bob, this is Alice!
Distorted Message =  This message is distorted--->0
Distorted message sent successfully
Message from Real Bob:  Hey Alice!
Distorted Message =  This message is distorted--->0
Original Message from Alice:  Are you reciving my messages?
This message from Alice is sent as it is
Distorted message sent successfully
Message from Real Bob:  Yes
This message from Bob is sent as it is
Closing TLSv1.3 and TCP Connection
root@trudy1:~#
```

Trudy distorting some messages

Credit Statement:

Tasks	Revanth Rokkam	Arkadeb Ghosh	Divya Pathak
Task 1	Did entirely		
Task 2	task2.pcap analysis	Collaborative work → TCP connection, TLS certificate updation in trust store by Divya and TLS establishment code by Arkadeb	
Task 3			Did entirely
Task 4		Did entirely	
Report	Collaborative work → Task 1 by Revanth, Task 2, 3, 4 collaboratively by Arkadeb and Divya		
Readme, Makefile, Pcap analysis	Collaborative work		
Bug Fixes	Collaborative work		

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: Revanth Rokkam, Arkadeb Ghosh, Divya Pathak

Date: 08-04-2022
Signature: RR, AG, DP

References used:

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

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: Revanth Rokkam, Divya Pathak, Arkadeb Ghosh
Date: 08/04/2022
Signature: RR, DP, AG