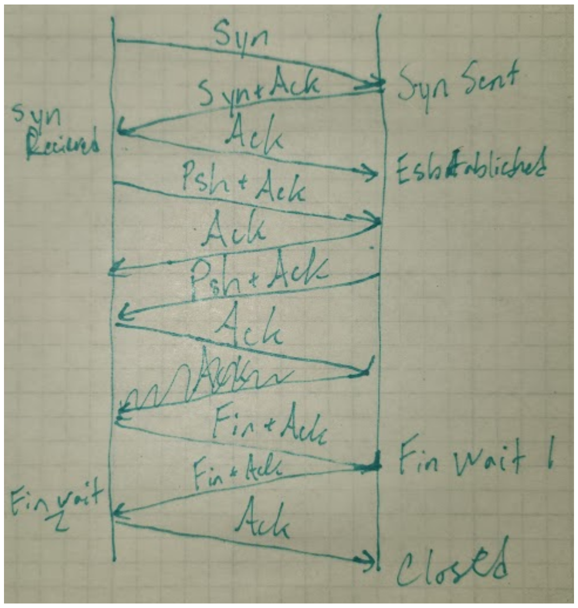
Ryan J Skelly CS/IT 725 Asn04 11/21/22

1. Describe the fundamental TCP protocol function of all the packets in the trace.
    a. 55261 -> 80, SYN, Connection Start
    b. 80 -> 55261, SYN + ACK, Connection Start Accept
    c. 55261 -> 80, ACK, Connection Established
    d. 55261 -> 80, PSH + ACK, Load URI(http://unh.edu/)
    e. 80 -> 55261, ACK, OK!
    f. 80 -> 55261, PSH + ACK, Error 301, Moved Resource
    g. 55261 -> 80, ACK, OK!
    h. 55261 -> 80, FIN + ACK, Close Connection
    i. 80 -> 55261, FIN + ACK, Close Connection
    j. 55261 -> 80, ACK, Connection Closed
2. Draw a sequence diagram showing the exchanged packets and for each identify its function (e.g., SYN, SYN+ACK, etc.). Label the vertical axes with TCP protocol states of both client and server.
    a. 

3. What are the absolute values of the initial sequence numbers of the connection (client to server and server to client)? It is OK to give hex values.
    a. 2137159159 from Client to server(SYN)
    b. 2701841408 from Server to client(SYN + ACK)
4. What was the total number of application payload bytes transmitted from the client to server and from the server to the client?
    a. 76+208=284
5. Estimate the round-trip time between the client and server. Give the packet pair(s) that you considered and why you chose them.
    a. For total connection roundtrip time, I'll be using the first SYN, SYN +ACK exchange against the final two FIN + ACK packets.

   b. Client=0.005212s
   c. Server=0.004822s
6. Are there ACK packets that cumulatively acknowledge multiple data packets?
   a. Yes, although not in this capture
7. Do the client and server agree on the SACK option? How?
   a. Agreed during the initial exchange
8. Does the trace shows the connection being closed?
   a. Yes, full close FIN+ACK->FIN+ACK->ACK

3.) NOTE: I went to https://www.cs.unh.edu/~cs725/assignments/a3.html, did Ctrl+S to save the file, and I saved it as the HTML only(by clicking the drop down for file type, and clicking "Webpage, HTML only")

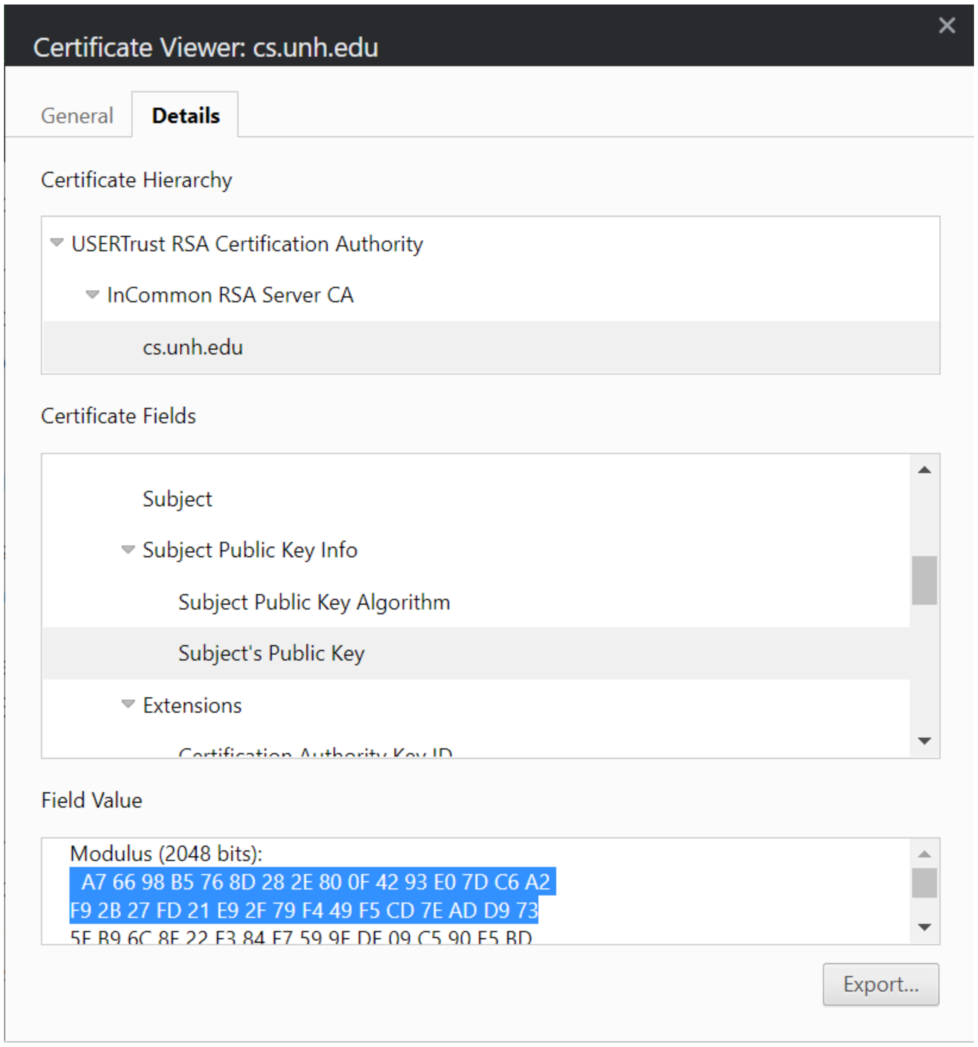
```
┌──(rskelly㉿LAPTOP-RLMT89M8)-[~]

└─$ md5sum a3.html

e99ccce353e9a5ba324202c6b98ff27a  a3.html

┌──(rskelly㉿LAPTOP-RLMT89M8)-[~]

└─$ sha512sum a3.html

767d9b25846b11e67148c3225bb06ba01e3310712d17241fbb2ecde613a9920900b58e
d3b339ff7a9b586f1cb340f2a712e094059186d9c4fdf75d7929f7a9cc  a3.html

┌──(rskelly㉿LAPTOP-RLMT89M8)-[~]

└─$
```

4.)First bytes of public key:

00:a7:66:98:b5:76:8d:28:2e:80:0f:42:93:e0:7d:

Issuer:

InCommon RSA Server CA



Certificate Hierarchy

- USERTrust RSA Certification Authority
  - InCommon RSA Server CA
    - cs.unh.edu

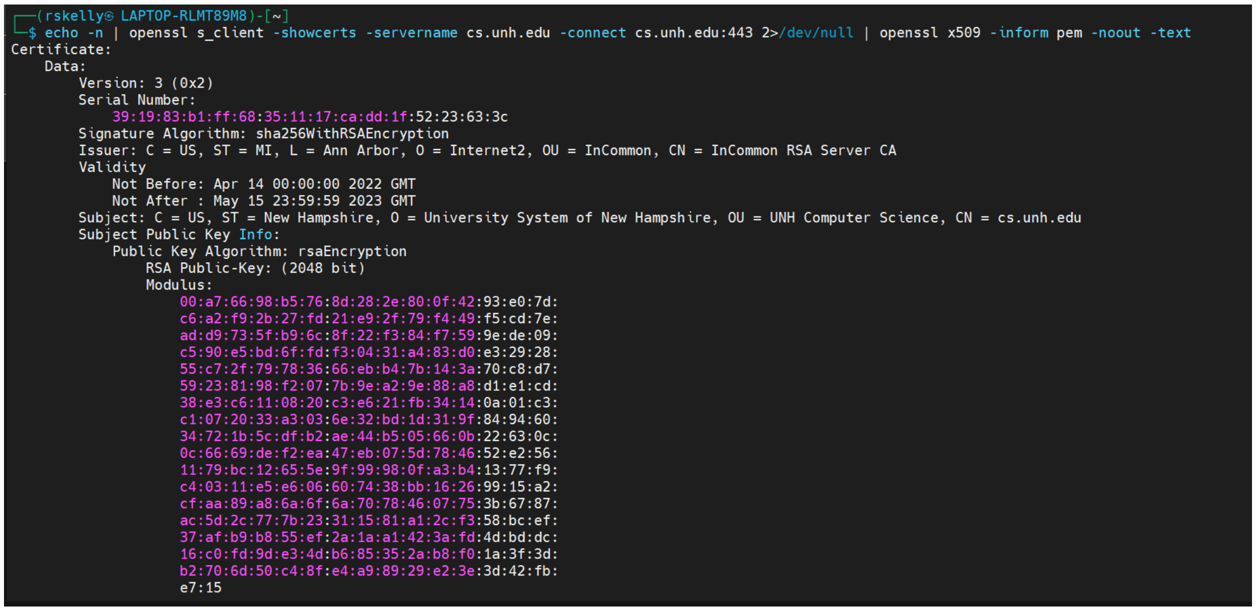
Certificate Fields

- Subject
- Subject Public Key Info
  - Subject Public Key Algorithm
  - Subject's Public Key
- Extensions

Field Value

```
Modulus (2048 bits):
 A7 66 98 B5 76 8D 28 2E 80 0F 42 93 E0 7D C6 A2
F9 2B 27 FD 21 E9 2F 79 F4 49 F5 CD 7E AD D9 73
5E B9 6C 8F 22 F3 84 F7 59 9F DF 09 C5 90 F5 BD
```

Using the openssl, you can find the same information, heres a screenshot of me grabbing it.



```
┌──(rskelly㉿ LAPTOP-RLMT89M8)-[~]
└─$ echo -n | openssl s_client -showcerts -servername cs.unh.edu -connect cs.unh.edu:443 2>/dev/null | openssl x509 -inform pem -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            39:19:83:b1:ff:68:35:11:17:ca:dd:1f:52:23:63:3c
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = US, ST = MI, L = Ann Arbor, O = Internet2, OU = InCommon, CN = InCommon RSA Server CA
        Validity
            Not Before: Apr 14 00:00:00 2022 GMT
            Not After : May 15 23:59:59 2023 GMT
        Subject: C = US, ST = New Hampshire, O = University System of New Hampshire, OU = UNH Computer Science, CN = cs.unh.edu
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:a7:66:98:b5:76:8d:28:2e:80:0f:42:93:e0:7d:
                    c6:a2:f9:2b:27:fd:21:e9:2f:79:f4:49:f5:cd:7e:
                    ad:d9:73:5f:b9:6c:8f:22:f3:84:f7:59:9e:de:09:
                    c5:90:e5:bd:6f:fd:f3:04:31:a4:83:d0:e3:29:28:
                    55:c7:2f:79:78:36:66:eb:b4:7b:14:3a:70:c8:d7:
                    59:23:81:98:f2:07:7b:9e:a2:9e:88:a8:d1:e1:cd:
                    38:e3:c6:11:08:20:c3:e6:21:fb:34:14:0a:01:c3:
                    c1:07:20:33:a3:03:6e:32:bd:1d:31:9f:84:94:60:
                    34:72:1b:5c:df:b2:ae:44:b5:05:66:0b:22:63:0c:
                    0c:66:69:de:f2:ea:47:eb:07:5d:78:46:52:e2:56:
                    11:79:bc:12:65:5e:9f:99:98:0f:a3:b4:13:77:f9:
                    c4:03:11:e5:e6:06:60:74:38:bb:16:26:99:15:a2:
                    cf:aa:89:a8:6a:6f:6a:70:78:46:07:75:3b:67:87:
                    ac:5d:2c:77:7b:23:31:15:81:a1:2c:f3:58:bc:ef:
                    37:af:b9:b8:55:ef:2a:1a:a1:42:3a:fd:4d:bd:dc:
                    16:c0:fd:9d:e3:4d:b6:85:35:2a:b8:f0:1a:3f:3d:
                    b2:70:6d:50:c4:8f:e4:a9:89:29:e2:3e:3d:42:fb:
                    e7:15
```

Program:
Flask based python script that serves the index.html at the ‘/’ file path, and a json object containing two time measures(sec, first once /time is requested, and the time of returning the two values) served at ‘/time’ path.