PUBLIC-KEY INFRASTRUCTURE

UFCFVN-30-M - Computer & Network Security

A Research Coursework

Prepared by

Kiran Kumar Mohanraj - 20054954

MSc Cyber Security

University of the West of England, Bristol

TABLE OF CONTENTS

1. LAB TASK

1.1 Task 1: RSA Key Cryptanalysis

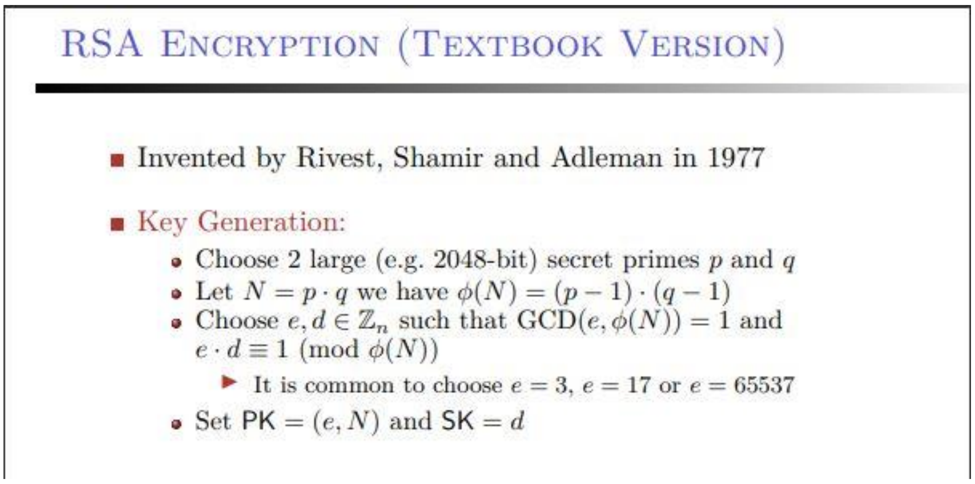
To find the RSA public key, the following details are provided:

N=0x8ee32e2bdc9175272b19d9d64667b3869411e62b090bb16c50cabf0b76353ebcda48f9ae02d0e27c75e2d5c0bff5c0bab5519609e96428fd86d968a27c619

e=0x11f87

p=0x39340499bcb48d0899db332321af58fc9d0c8424d5992fe255da980c4a2e117

Figure 1 show the procedure for RSA Encryption and how public keys are generated.

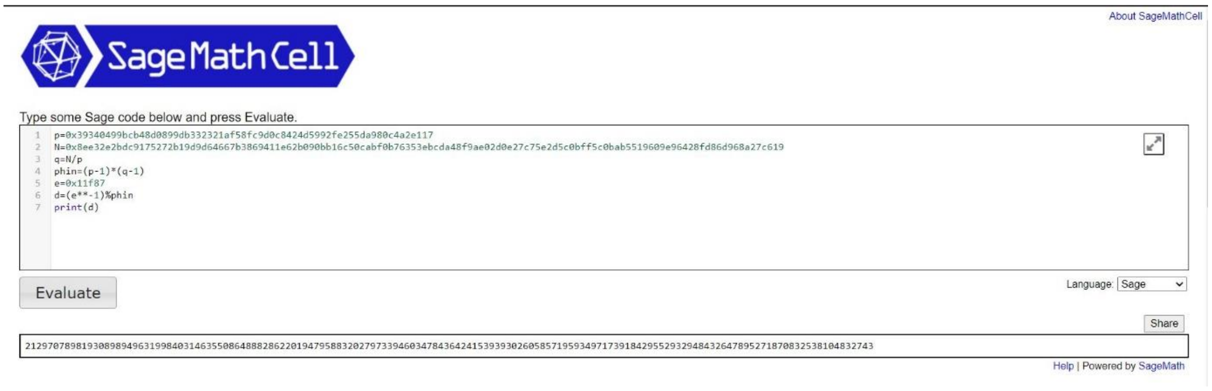


RSA Encryption (Textbook Version)

- Invented by Rivest, Shamir and Adleman in 1977
- Key Generation:
  - Choose 2 large (e.g. 2048-bit) secret primes p and q
  - Let $N = p \cdot q$ we have $\phi(N) = (p-1) \cdot (q-1)$
  - Choose $e, d \in \mathbb{Z}_n$ such that $\mathrm{GCD}(e, \phi(N)) = 1$ and $e \cdot d \equiv 1 \pmod{\phi(N)}$
    - It is common to choose $e = 3$, $e = 17$ or $e = 65537$
  - Set $\mathsf{PK} = (e, N)$ and $\mathsf{SK} = d$

Figure 1 RSA Encryption

For this task, we use SageMathCell for carrying out the mathematical calculation. Using the above shown formulas, we derive the value of d (public key) as shown in Figure 2.



Type some Sage code below and press Evaluate.

```
p=0x39340499bcb48d0899db332321af58fc9d0c8424d5992fe255da980c4a2e117
N=0x8ee32e2bdc9175272b19d9d64667b3869411e62b090bb16c50cabf0b76353ebcda48f9ae02d0e27c75e2d5c0bff5c0bab5519609e96428fd86d968a27c619
q=N/p
phin=(p-1)*(q-1)
e=0x11f87
d=(e**-1)%phin
print(d)
```

```
212970789819308989496319984031463550864888286220194795883202797339460347843642415393930260585719593497173918429552932948432647895271870832538104832743
```

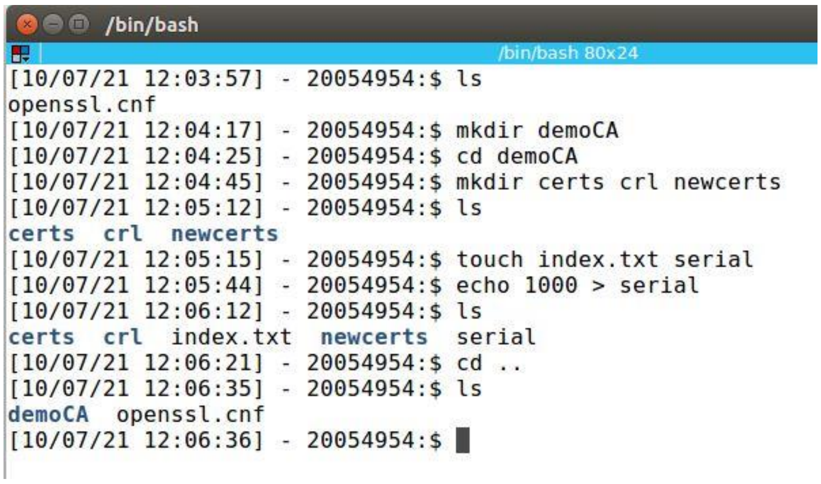
Figure 2 Calculations for finding RSA public key

The RSA public key is:

D=212970789819308989496319984031463550864888286220194795883202797339460347843642415393930260585719593497173918429552932948432647895271870832538104832743 (in decimals)

1.2 Task 2: Creating a Certificate Authority (CA)

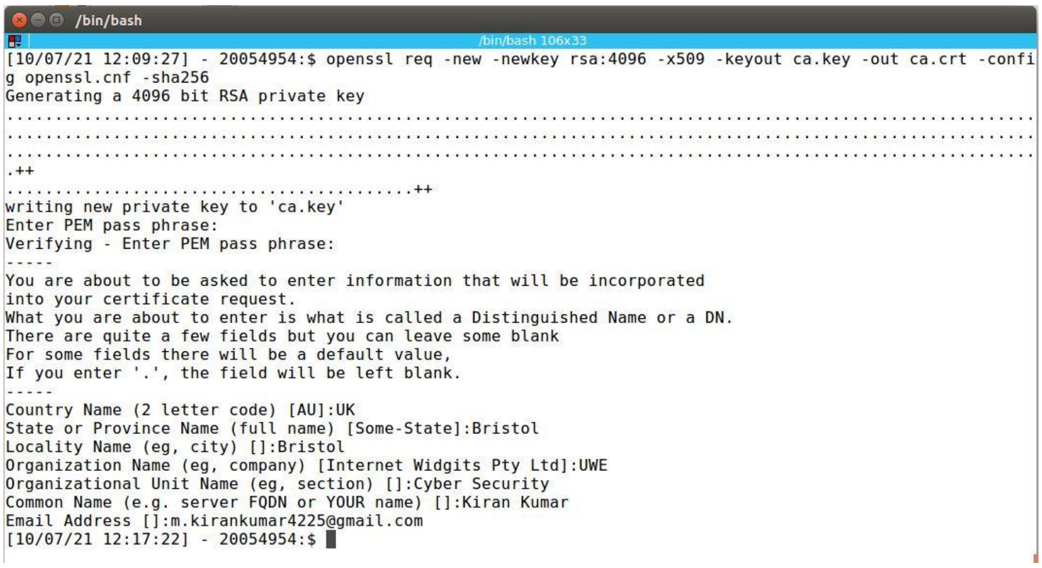
Before creating a Certificate Authority, we need to setup the configuration file (openssl.cnf). To use this file, we need to create some directories and sub-directories. These steps are illustrated in Figure 3.



```
[10/07/21 12:03:57] - 20054954:$ ls
openssl.cnf
[10/07/21 12:04:17] - 20054954:$ mkdir demoCA
[10/07/21 12:04:25] - 20054954:$ cd demoCA
[10/07/21 12:04:45] - 20054954:$ mkdir certs crl newcerts
[10/07/21 12:05:12] - 20054954:$ ls
certs  crl  newcerts
[10/07/21 12:05:15] - 20054954:$ touch index.txt serial
[10/07/21 12:05:44] - 20054954:$ echo 1000 > serial
[10/07/21 12:06:12] - 20054954:$ ls
certs  crl  index.txt  newcerts  serial
[10/07/21 12:06:21] - 20054954:$ cd ..
[10/07/21 12:06:35] - 20054954:$ ls
demoCA  openssl.cnf
[10/07/21 12:06:36] - 20054954:$
```

Figure 3 Configuration file setup

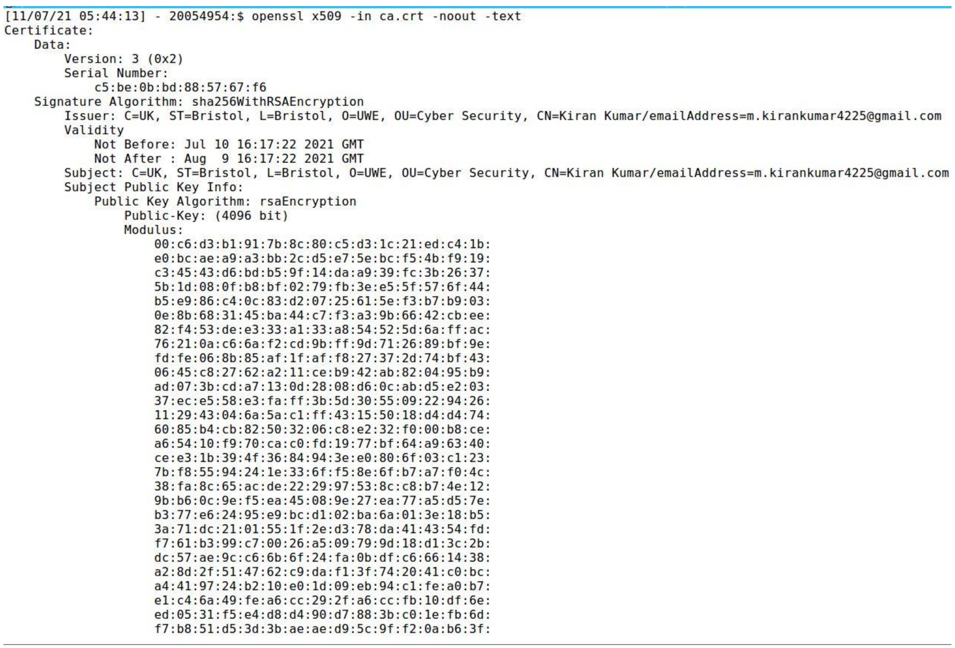
Once the setup is done, we create a self-signed certificate for our CA. Figure 4 shows the command for self signing a certificate. The outputs of this command is stored in ca.key and ca.crt files. The ca.key stores the private key of the CA and ca.crt stores the public-key certificate. The details provided for self-signing a certificate are shown in Figure 4.



```
[10/07/21 12:09:27] - 20054954:$ openssl req -new -newkey rsa:4096 -x509 -keyout ca.key -out ca.crt -confi
g openssl.cnf -sha256
Generating a 4096 bit RSA private key
..........................................................................................................
..........................................................................................................
..........................................................................................................
.++
.........................................++
writing new private key to 'ca.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:UK
State or Province Name (full name) [Some-State]:Bristol
Locality Name (eg, city) []:Bristol
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UWE
Organizational Unit Name (eg, section) []:Cyber Security
Common Name (e.g. server FQDN or YOUR name) []:Kiran Kumar
Email Address []:m.kirankumar4225@gmail.com
[10/07/21 12:17:22] - 20054954:$
```

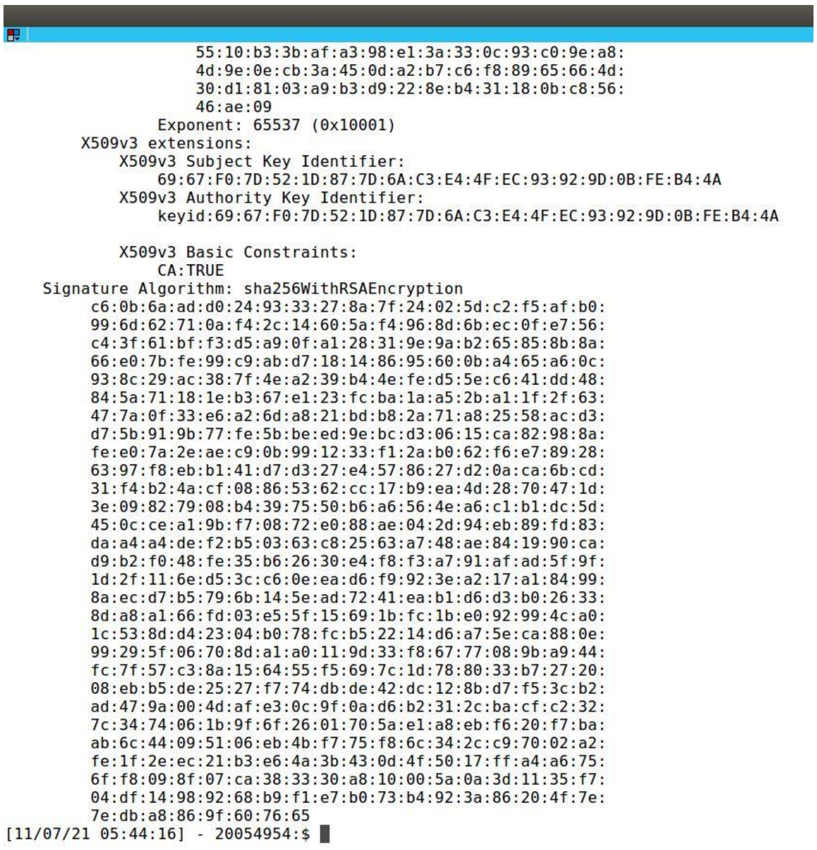
Figure 4 Self-signed Certificate creation for CA

Figure 5 and 6 shows the content of the certificate. The subject and issuer are same which proves that it is a self-signed certificate. In Basic Constraints, the CA is set TRUE, this shows that this certificate can sign another certificate. This proves that Certificate Authority has been created successfully.



```
[11/07/21 05:44:13] - 20054954:$ openssl x509 -in ca.crt -noout -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            c5:be:0b:bd:88:57:67:f6
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=UK, ST=Bristol, L=Bristol, O=UWE, OU=Cyber Security, CN=Kiran Kumar/emailAddress=m.kirankumar4225@gmail.com
        Validity
            Not Before: Jul 10 16:17:22 2021 GMT
            Not After : Aug  9 16:17:22 2021 GMT
        Subject: C=UK, ST=Bristol, L=Bristol, O=UWE, OU=Cyber Security, CN=Kiran Kumar/emailAddress=m.kirankumar4225@gmail.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:c6:d3:b1:91:7b:8c:80:c5:d3:1c:21:ed:c4:1b:
                    e0:bc:ae:a9:a3:bb:2c:d5:e7:5e:bc:f5:4b:f9:19:
                    c3:45:43:d6:bd:b5:9f:14:da:a9:39:fc:3b:26:37:
                    5b:1d:08:0f:b8:bf:02:79:fb:3e:e5:5f:57:6f:44:
                    b5:e9:86:c4:0c:83:d2:07:25:61:5e:f3:b7:b9:03:
                    0e:8b:68:31:45:ba:44:c7:f3:a3:9b:66:42:cb:ee:
                    82:f4:53:de:e3:33:a1:33:a8:54:52:5d:6a:ff:ac:
                    76:21:0a:c6:6a:f2:cd:9b:ff:9d:71:26:89:bf:9e:
                    fd:fe:06:8b:85:af:1f:af:f8:27:37:2d:74:bf:43:
                    06:45:c8:27:62:a2:11:ce:b9:42:ab:82:04:95:b9:
                    ad:07:3b:cd:a7:13:0d:28:08:d6:0c:ab:d5:e2:03:
                    37:ec:e5:58:e3:fa:ff:3b:5d:30:55:09:22:94:26:
                    11:29:43:04:6a:5a:c1:ff:43:15:50:18:d4:d4:74:
                    60:85:b4:cb:82:50:32:06:c8:e2:32:f0:00:b8:ce:
                    a6:54:10:f9:70:ca:c0:fd:19:77:bf:64:a9:63:40:
                    ce:e3:1b:39:4f:36:84:94:3e:e0:80:6f:03:c1:23:
                    7b:f8:55:94:24:1e:33:6f:f5:8e:6f:b7:a7:f0:4c:
                    38:fa:8c:65:ac:de:22:29:97:53:8c:c8:b7:4e:12:
                    9b:b6:0c:9e:f5:ea:45:08:9e:27:ea:77:a5:d5:7e:
                    b3:77:e6:24:95:e9:bc:d1:02:ba:6a:01:3e:18:b5:
                    3a:71:dc:21:01:55:1f:2e:d3:78:da:41:43:54:fd:
                    f7:61:b3:99:c7:00:26:a5:09:79:9d:18:d1:3c:2b:
                    dc:57:ae:9c:c6:6b:6f:24:fa:0b:df:c6:66:14:38:
                    a2:8d:2f:51:47:62:c9:da:f1:3f:74:20:41:c0:bc:
                    a4:41:97:24:b2:10:e0:1d:09:eb:94:c1:fe:a0:b7:
                    e1:c4:6a:49:fe:a6:cc:29:2f:a6:cc:fb:10:df:6e:
                    ed:05:31:f5:e4:d8:d4:90:d7:88:3b:c0:1e:fb:6d:
                    f7:b8:51:d5:3d:3b:ae:ae:d9:5c:9f:f2:0a:b6:3f:
```

Figure 5 Self-signed Certificate

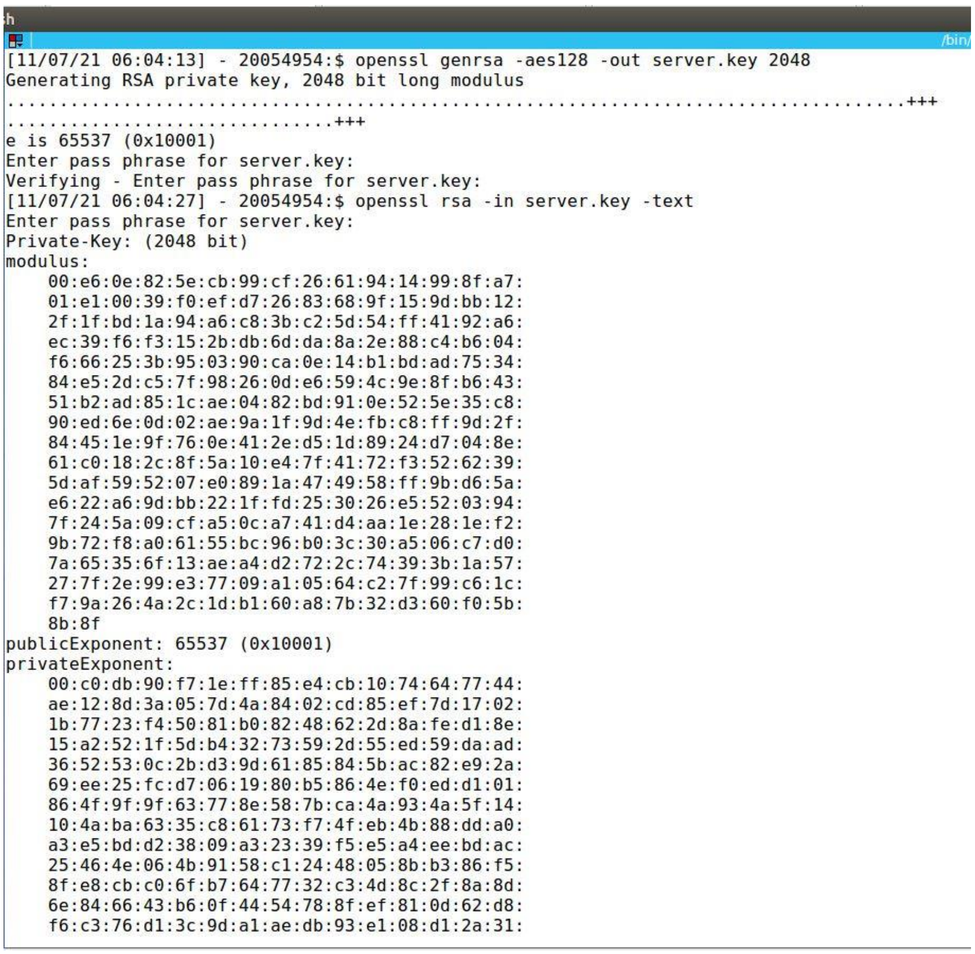


```
                    55:10:b3:3b:af:a3:98:e1:3a:33:0c:93:c0:9e:a8:
                    4d:9e:0e:cb:3a:45:0d:a2:b7:c6:f8:89:65:66:4d:
                    30:d1:81:03:a9:b3:d9:22:8e:b4:31:18:0b:c8:56:
                    46:ae:09
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                69:67:F0:7D:52:1D:87:7D:6A:C3:E4:4F:EC:93:92:9D:0B:FE:B4:4A
            X509v3 Authority Key Identifier:
                keyid:69:67:F0:7D:52:1D:87:7D:6A:C3:E4:4F:EC:93:92:9D:0B:FE:B4:4A

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         c6:0b:6a:ad:d0:24:93:33:27:8a:7f:24:02:5d:c2:f5:af:b0:
         99:6d:62:71:0a:f4:2c:14:60:5a:f4:96:8d:6b:ec:0f:e7:56:
         c4:3f:61:bf:f3:d5:a9:0f:a1:28:31:9e:9a:b2:65:85:8b:8a:
         66:e0:7b:fe:99:c9:ab:d7:18:14:86:95:60:0b:a4:65:a6:0c:
         93:8c:29:ac:38:7f:4e:a2:39:b4:4e:fe:d5:5e:c6:41:dd:48:
         84:5a:71:18:1e:b3:67:e1:23:fc:ba:1a:a5:2b:a1:1f:2f:63:
         47:7a:0f:33:e6:a2:6d:a8:21:bd:b8:2a:71:a8:25:58:ac:d3:
         d7:5b:91:9b:77:fe:5b:be:ed:9e:bc:d3:06:15:ca:82:98:8a:
         fe:e0:7a:2e:ae:c9:0b:99:12:33:f1:2a:b0:62:f6:e7:89:28:
         63:97:f8:eb:b1:41:d7:d3:27:e4:57:86:27:d2:0a:ca:6b:cd:
         31:f4:b2:4a:cf:08:86:53:62:cc:17:b9:ea:4d:28:70:47:1d:
         3e:09:82:79:08:b4:39:75:50:b6:a6:56:4e:a6:c1:b1:dc:5d:
         45:0c:ce:a1:9b:f7:08:72:e0:88:ae:04:2d:94:eb:89:fd:83:
         da:a4:a4:de:f2:b5:03:63:c8:25:63:a7:48:ae:84:19:90:ca:
         d9:b2:f0:48:fe:35:b6:26:30:e4:f8:f3:a7:91:af:ad:5f:9f:
         1d:2f:11:6e:d5:3c:c6:0e:ea:d6:f9:92:3e:a2:17:a1:84:99:
         8a:ec:d7:b5:79:6b:14:5e:ad:72:41:ea:b1:d6:d3:b0:26:33:
         8d:a8:a1:66:fd:03:e5:5f:15:69:1b:fc:1b:e0:92:99:4c:a0:
         1c:53:8d:d4:23:04:b0:78:fc:b5:22:14:d6:a7:5e:ca:88:0e:
         99:29:5f:06:70:8d:a1:a0:11:9d:33:f8:67:77:08:9b:a9:44:
         fc:7f:57:c3:8a:15:64:55:f5:69:7c:1d:78:80:33:b7:27:20:
         08:eb:b5:de:25:27:f7:74:db:de:42:dc:12:8b:d7:f5:3c:b2:
         ad:47:9a:00:4d:af:e3:0c:9f:0a:d6:b2:31:2c:ba:cf:c2:32:
         7c:34:74:06:1b:9f:6f:26:01:70:5a:e1:a8:eb:f6:20:f7:ba:
         ab:6c:44:09:51:06:eb:4b:f7:75:f8:6c:34:2c:c9:70:02:a2:
         fe:1f:2e:ec:21:b3:e6:4a:3b:43:0d:4f:50:17:ff:a4:a6:75:
         6f:f8:09:8f:07:ca:38:33:30:a8:10:00:5a:0a:3d:11:35:f7:
         04:df:14:98:92:68:b9:f1:e7:b0:73:b4:92:3a:86:20:4f:7e:
         7e:db:a8:86:9f:60:76:65
[11/07/21 05:44:16] - 20054954:$
```

Figure 6 Self-signed Certificate

1.3 Task 3: Using your CA to Issue Certificates

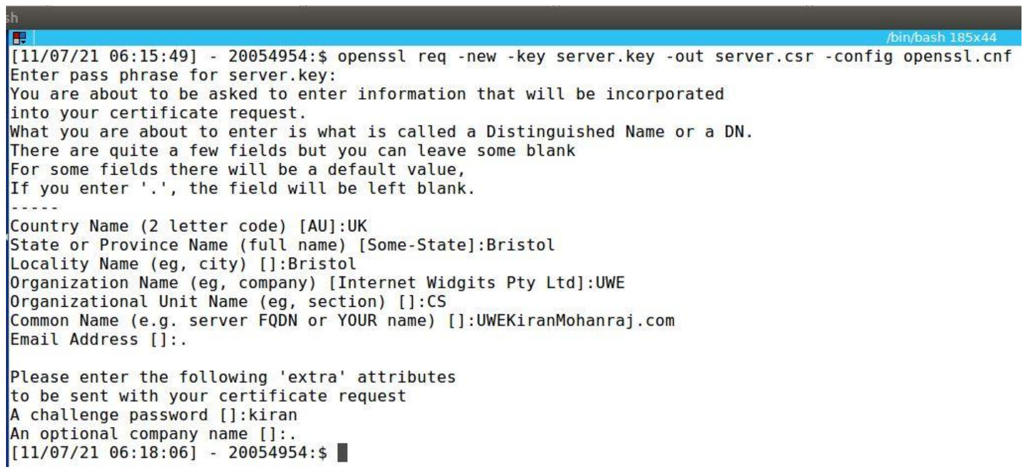
Firstly, we need to generate RSA public-private key pairs. Figure 7 shows the commands used for key pair generation. To view the key pair, we use the command “openssl rsa -in server.key -text”.



```
[11/07/21 06:04:13] - 20054954:$ openssl genrsa -aes128 -out server.key 2048
Generating RSA private key, 2048 bit long modulus
..................................................................................+++
...............................+++
e is 65537 (0x10001)
Enter pass phrase for server.key:
Verifying - Enter pass phrase for server.key:
[11/07/21 06:04:27] - 20054954:$ openssl rsa -in server.key -text
Enter pass phrase for server.key:
Private-Key: (2048 bit)
modulus:
    00:e6:0e:82:5e:cb:99:cf:26:61:94:14:99:8f:a7:
    01:e1:00:39:f0:ef:d7:26:83:68:9f:15:9d:bb:12:
    2f:1f:bd:1a:94:a6:c8:3b:c2:5d:54:ff:41:92:a6:
    ec:39:f6:f3:15:2b:db:6d:da:8a:2e:88:c4:b6:04:
    f6:66:25:3b:95:03:90:ca:0e:14:b1:bd:ad:75:34:
    84:e5:2d:c5:7f:98:26:0d:e6:59:4c:9e:8f:b6:43:
    51:b2:ad:85:1c:ae:04:82:bd:91:0e:52:5e:35:c8:
    90:ed:6e:0d:02:ae:9a:1f:9d:4e:fb:c8:ff:9d:2f:
    84:45:1e:9f:76:0e:41:2e:d5:1d:89:24:d7:04:8e:
    61:c0:18:2c:8f:5a:10:e4:7f:41:72:f3:52:62:39:
    5d:af:59:52:07:e0:89:1a:47:49:58:ff:9b:d6:5a:
    e6:22:a6:9d:bb:22:1f:fd:25:30:26:e5:52:03:94:
    7f:24:5a:09:cf:a5:0c:a7:41:d4:aa:1e:28:1e:f2:
    9b:72:f8:a0:61:55:bc:96:b0:3c:30:a5:06:c7:d0:
    7a:65:35:6f:13:ae:a4:d2:72:2c:74:39:3b:1a:57:
    27:7f:2e:99:e3:77:09:a1:05:64:c2:7f:99:c6:1c:
    f7:9a:26:4a:2c:1d:b1:60:a8:7b:32:d3:60:f0:5b:
    8b:8f
publicExponent: 65537 (0x10001)
privateExponent:
    00:c0:db:90:f7:1e:ff:85:e4:cb:10:74:64:77:44:
    ae:12:8d:3a:05:7d:4a:84:02:cd:85:ef:7d:17:02:
    1b:77:23:f4:50:81:b0:82:48:62:2d:8a:fe:d1:8e:
    15:a2:52:1f:5d:b4:32:73:59:2d:55:ed:59:da:ad:
    36:52:53:0c:2b:d3:9d:61:85:84:5b:ac:82:e9:2a:
    69:ee:25:fc:d7:06:19:80:b5:86:4e:f0:ed:d1:01:
    86:4f:9f:9f:63:77:8e:58:7b:ca:4a:93:4a:5f:14:
    10:4a:ba:63:35:c8:61:73:f7:4f:eb:4b:88:dd:a0:
    a3:e5:bd:d2:38:09:a3:23:39:f5:e5:a4:ee:bd:ac:
    25:46:4e:06:4b:91:58:c1:24:48:05:8b:b3:86:f5:
    8f:e8:cb:c0:6f:b7:64:77:32:c3:4d:8c:2f:8a:8d:
    6e:84:66:43:b6:0f:44:54:78:8f:ef:81:0d:62:d8:
    f6:c3:76:d1:3c:9d:a1:ae:db:93:e1:08:d1:2a:31:
```

Figure 7 RSA key pair generation

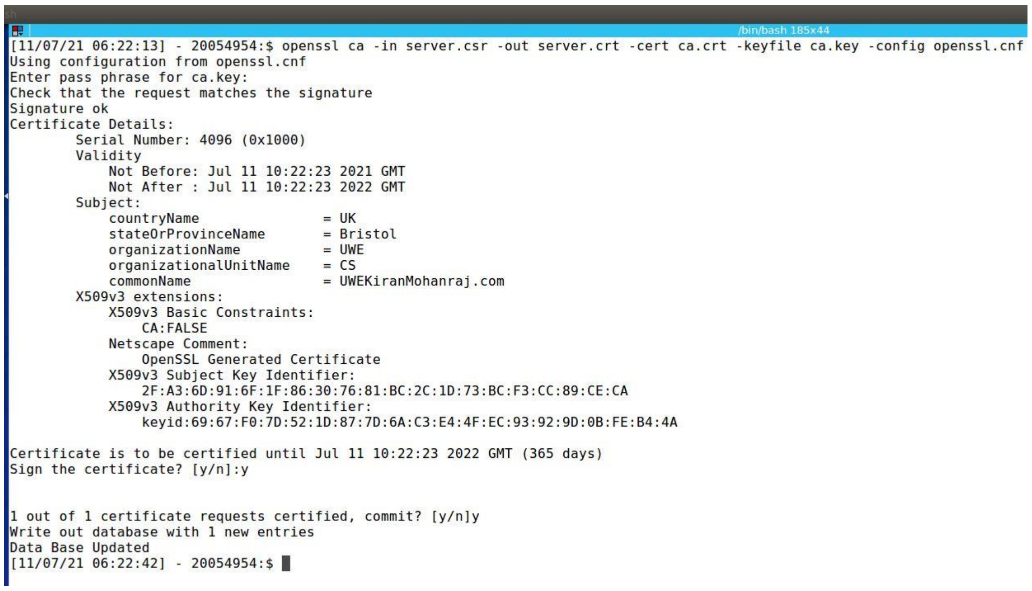
Now, we create Certificate Signing Request (CSR) with the previously generated server key pair. This CSR is then sent to the CA to generate a certificate for the key and company name.



```
[11/07/21 06:15:49] - 20054954:$ openssl req -new -key server.key -out server.csr -config openssl.cnf
Enter pass phrase for server.key:
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]:UK
State or Province Name (full name) [Some-State]:Bristol
Locality Name (eg, city) []:Bristol
Organization Name (eg, company) [Internet Widgits Pty Ltd]:UWE
Organizational Unit Name (eg, section) []:CS
Common Name (e.g. server FQDN or YOUR name) []:UWEKiranMohanraj.com
Email Address []:.

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:kiran
An optional company name []:.
[11/07/21 06:18:06] - 20054954:$
```

Figure 8 Creating CSR

Figure 9 shows the commands for issuing the certificate where the previously generated CSR is converted into X509 certificate using the CA's private key and public-key certificate.

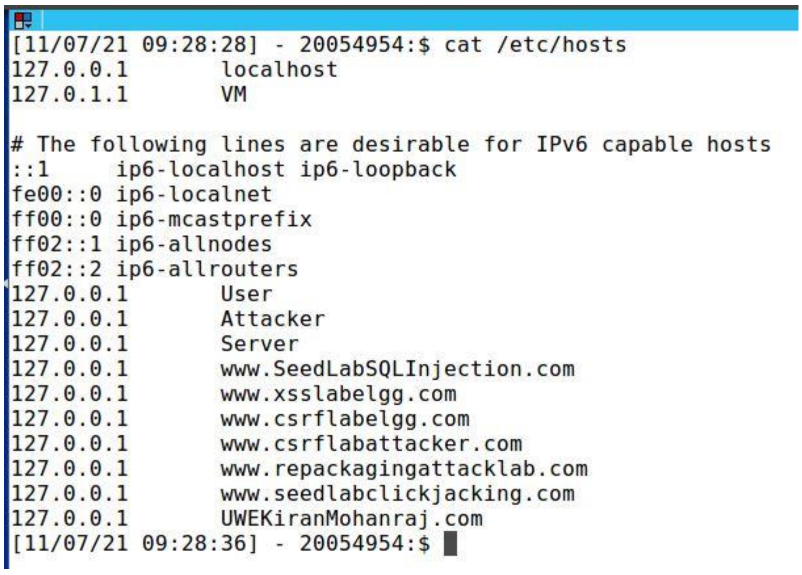


```
[11/07/21 06:22:13] - 20054954:$ openssl ca -in server.csr -out server.crt -cert ca.crt -keyfile ca.key -config openssl.cnf
Using configuration from openssl.cnf
Enter pass phrase for ca.key:
Check that the request matches the signature
Signature ok
Certificate Details:
        Serial Number: 4096 (0x1000)
        Validity
            Not Before: Jul 11 10:22:23 2021 GMT
            Not After : Jul 11 10:22:23 2022 GMT
        Subject:
            countryName               = UK
            stateOrProvinceName       = Bristol
            organizationName          = UWE
            organizationalUnitName    = CS
            commonName                = UWEKiranMohanraj.com
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Comment:
                OpenSSL Generated Certificate
            X509v3 Subject Key Identifier:
                2F:A3:6D:91:6F:1F:86:30:76:81:BC:2C:1D:73:BC:F3:CC:89:CE:CA
            X509v3 Authority Key Identifier:
                keyid:69:67:F0:7D:52:1D:87:7D:6A:C3:E4:4F:EC:93:92:9D:0B:FE:B4:4A

Certificate is to be certified until Jul 11 10:22:23 2022 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated
[11/07/21 06:22:42] - 20054954:$
```

Figure 9 Issuing X509 Certificate

1.4 Task 4: Deploying Certificates in OpenSSL HTTPS Server

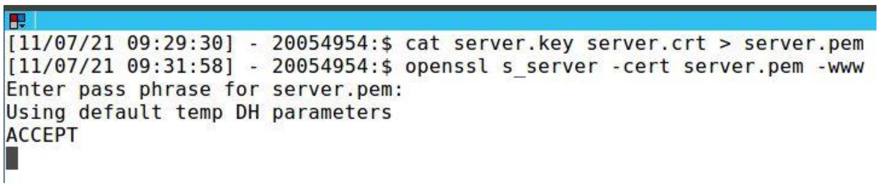
Firstly, the DNS should be configured. We need to add our entry to /etc/hosts file so that the machine recognizes the our website without changing the DNS. This entry maps the hostname to the localhost. Figure 10 shows the demonstration of the above steps.



```
[11/07/21 09:28:28] - 20054954:$ cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       VM

# The following lines are desirable for IPv6 capable hosts
::1     ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
127.0.0.1       UWEKiranMohanraj.com
[11/07/21 09:28:36] - 20054954:$ 
```

Figure 10 Configuring DNS for our hostname

For configuring the web server, we combine the key and certificate into a single file and then launch a web server using openssl s_server command. Figure 11 shows that the server is in ACCEPT state, indicating server is running.

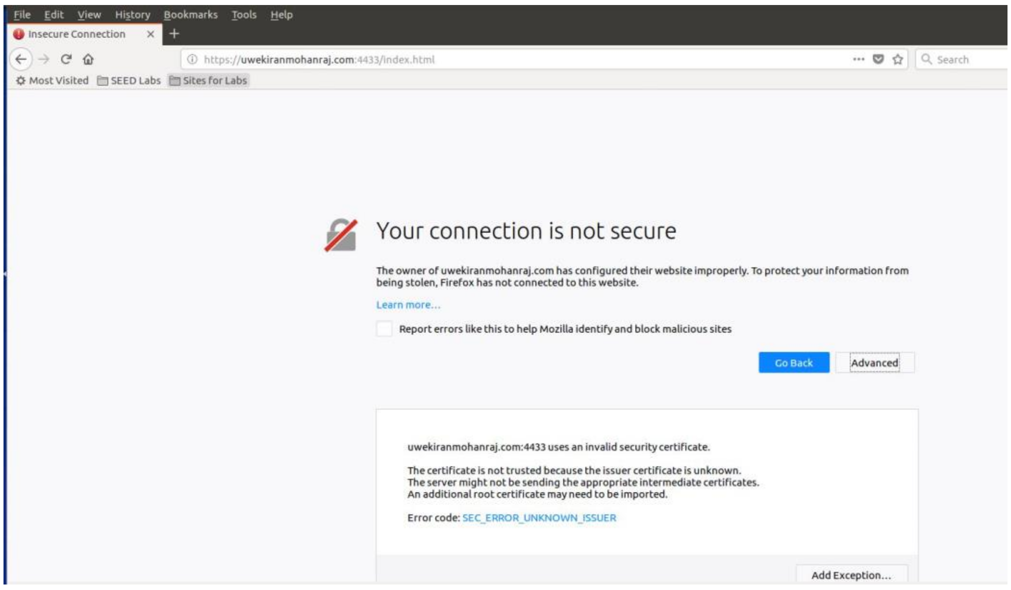


```
[11/07/21 09:29:30] - 20054954:$ cat server.key server.crt > server.pem
[11/07/21 09:31:58] - 20054954:$ openssl s_server -cert server.pem -www
Enter pass phrase for server.pem:
Using default temp DH parameters
ACCEPT
```

Figure 11 Configuring web server

The port 4433 is the default port which the server can listen. The URL for accessing the server is: www.UWEKiranMohanraj.com:4433/

When we load the URL, we could see an error, which says connection insecure, is displayed in the web browser. In Advanced options, we could see a message that our URL uses invalid security certificates. This is because, the root certificate which we created is not recognised by browser, thus invalidating our certificate.



Your connection is not secure

The owner of uwekiranmohanraj.com has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Report errors like this to help Mozilla identify and block malicious sites

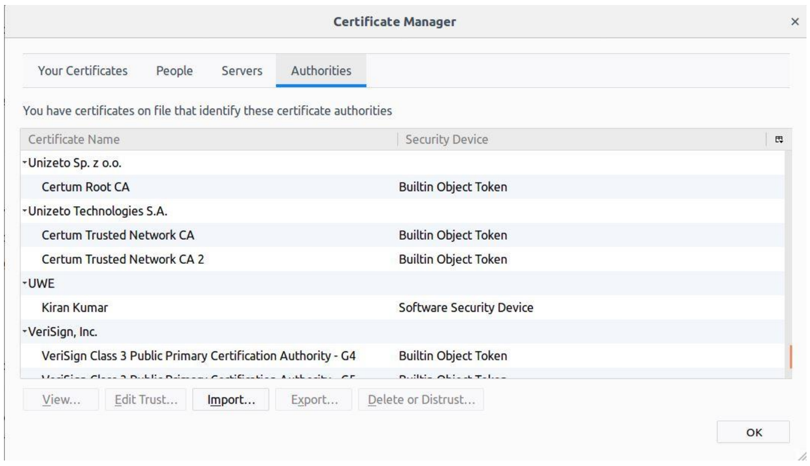
uwekiranmohanraj.com:4433 uses an invalid security certificate.

The certificate is not trusted because the issuer certificate is unknown.
The server might not be sending the appropriate intermediate certificates.
An additional root certificate may need to be imported.

Error code: SEC_ERROR_UNKNOWN_ISSUER

Figure 12 Accessing the web server using our URL

In order to overcome this error, we manually add our certificate to the browser by navigating to certificate manager and importing ca.crt. Figure 13 shows the our root CA has been added as trusted certificates.

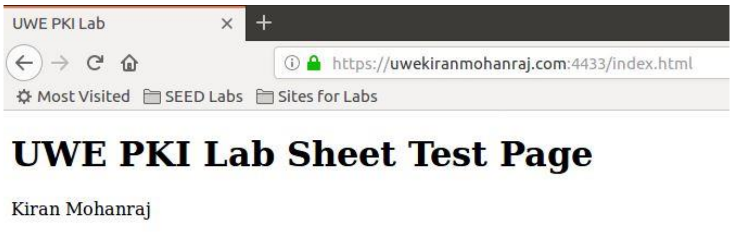


You have certificates on file that identify these certificate authorities

Certificate Name	Security Device
-Unizeto Sp. z o.o.	
Certum Root CA	Builtin Object Token
-Unizeto Technologies S.A.	
Certum Trusted Network CA	Builtin Object Token
Certum Trusted Network CA 2	Builtin Object Token
-UWE	
Kiran Kumar	Software Security Device
-VeriSign, Inc.	
VeriSign Class 3 Public Primary Certification Authority - G4	Builtin Object Token

Figure 13 Trusted Certificates of Firefox

Once this setup is done, we try to load our URL in the brower. We could see our index.html being loaded and contents of it has been displayed (as shown in Figure 14).



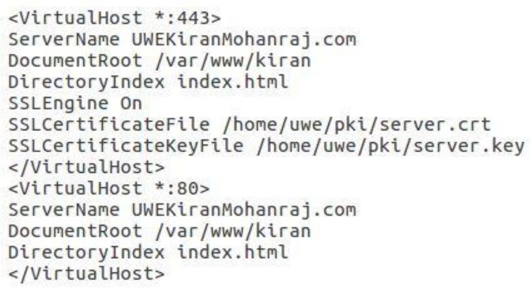
UWE PKI Lab Sheet Test Page

Kiran Mohanraj

Figure 14 Successfully accessing our web server

1.5 Task 5: Deploying Certificates in Apache HTTPS Server

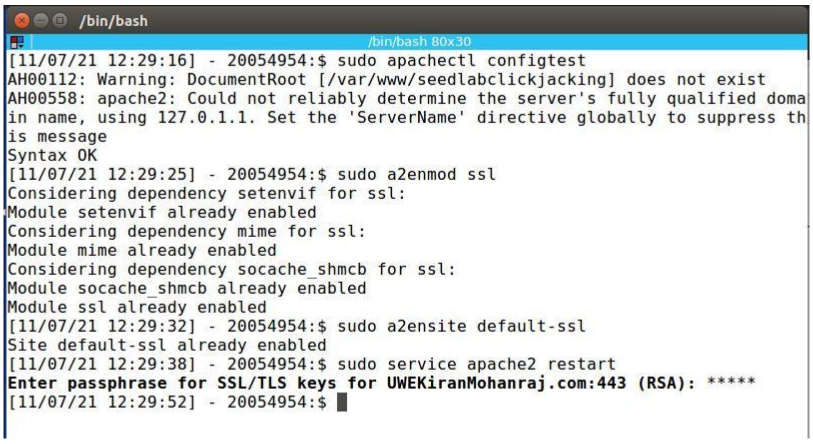
For this we use Apache web server for deploying the certificates. Firstly, we add a VirtualHost entry to the Apache configuration file (default-ssl.conf). Figure 15 shows our entry added to the configuration. The ServerName entry specifies the name of the website, while the DocumentRoot entry specifies where the files for the website are stored



```
<VirtualHost *:443>
ServerName UWEKiranMohanraj.com
DocumentRoot /var/www/kiran
DirectoryIndex index.html
SSLEngine On
SSLCertificateFile /home/uwe/pki/server.crt
SSLCertificateKeyFile /home/uwe/pki/server.key
</VirtualHost>
<VirtualHost *:80>
ServerName UWEKiranMohanraj.com
DocumentRoot /var/www/kiran
DirectoryIndex index.html
</VirtualHost>
```

Figure 15 VirtualHost entry

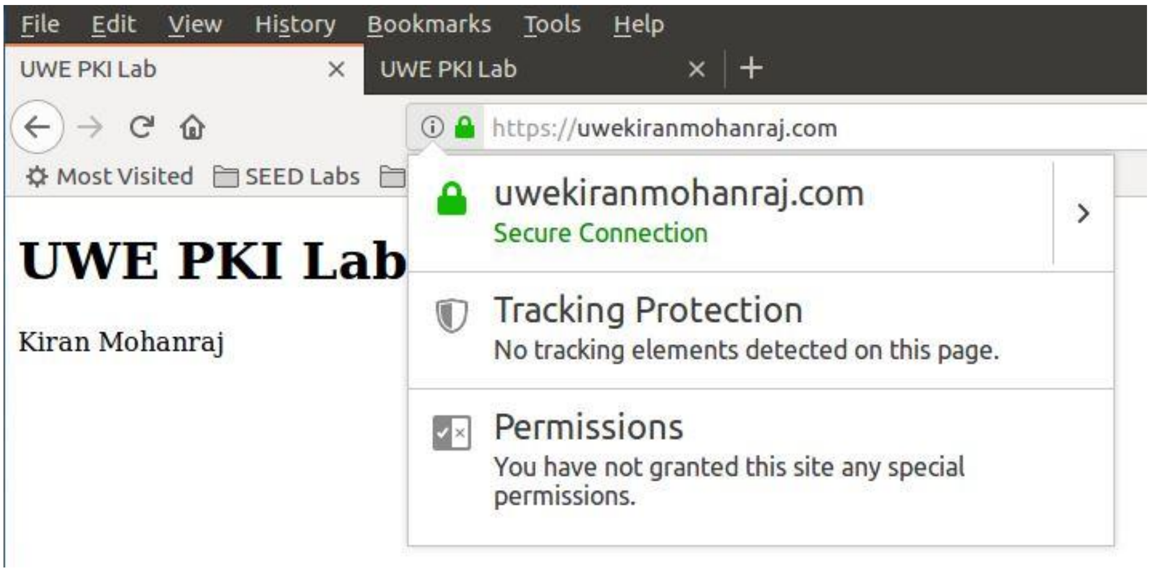
Once the setup is done, we run series of commands for starting the apache server. Firstly, the confirguration file is tested, then the SSL module is enabled, finnaly the apache server is configured for HTTPS and restarted. Figure 16 shows the series of commands executed for doing these steps.



```
[11/07/21 12:29:16] - 20054954:$ sudo apachectl configtest
AH00112: Warning: DocumentRoot [/var/www/seedlabclickjacking] does not exist
AH00558: apache2: Could not reliably determine the server's fully qualified doma
in name, using 127.0.1.1. Set the 'ServerName' directive globally to suppress th
is message
Syntax OK
[11/07/21 12:29:25] - 20054954:$ sudo a2enmod ssl
Considering dependency setenvif for ssl:
Module setenvif already enabled
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled
[11/07/21 12:29:32] - 20054954:$ sudo a2ensite default-ssl
Site default-ssl already enabled
[11/07/21 12:29:38] - 20054954:$ sudo service apache2 restart
Enter passphrase for SSL/TLS keys for UWEKiranMohanraj.com:443 (RSA): *****
[11/07/21 12:29:52] - 20054954:$
```

Figure 16 Configuring Apache server

Finally, the web browser is launched for loading the 2 URLs. Figure 17 shows the loading of https://uwekiranmohanraj.com and Figure 18 shows the loading of http://uwekiranmohanraj.com. The difference that we can notice is that URL with https shows a secure connection whereas the http shows connection insecure. The https URL is secure because the data transfered from the server to the browser is encrypted using SSL, thus creating a secure encrypted connection.


File Edit View History Bookmarks Tools Help

UWE PKI Lab | UWE PKI Lab

https://uwekiranmohanraj.com

Most Visited SEED Labs

UWE PKI Lab

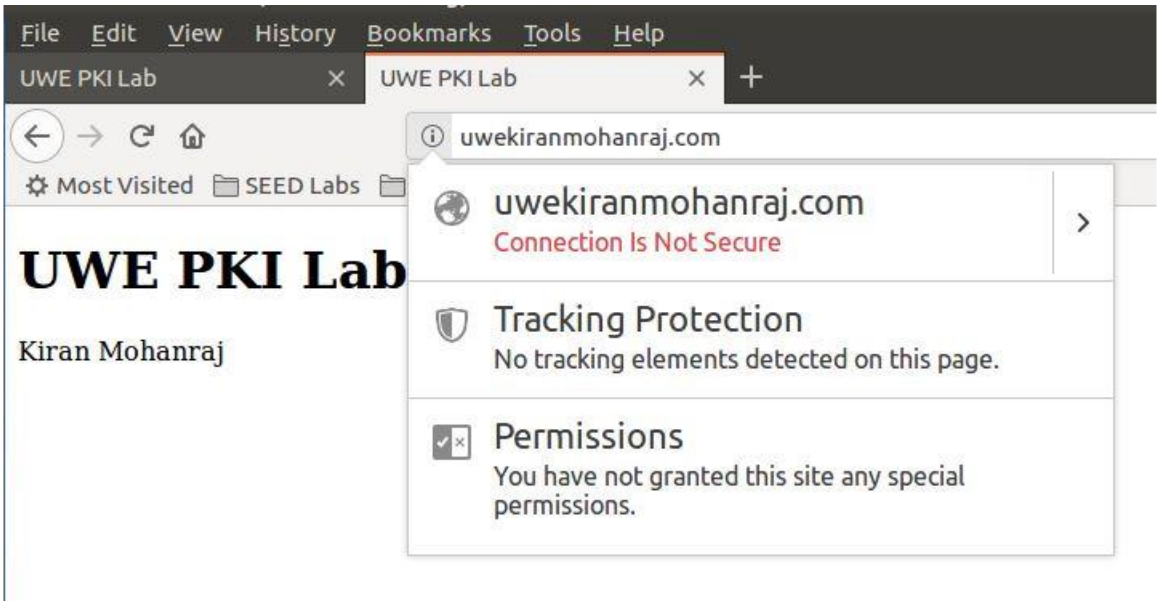
Kiran Mohanraj

uwekiranmohanraj.com
Secure Connection

Tracking Protection
No tracking elements detected on this page.

Permissions
You have not granted this site any special permissions.

Figure 17 HTTPS secure connection


File Edit View History Bookmarks Tools Help

UWE PKI Lab | UWE PKI Lab

uwekiranmohanraj.com

Most Visited SEED Labs

UWE PKI Lab

Kiran Mohanraj

uwekiranmohanraj.com
Connection Is Not Secure

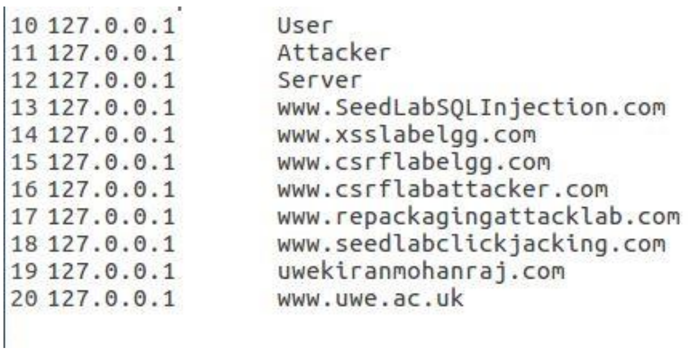
Tracking Protection
No tracking elements detected on this page.

Permissions
You have not granted this site any special permissions.

Figure 18 HTTP insecure connection

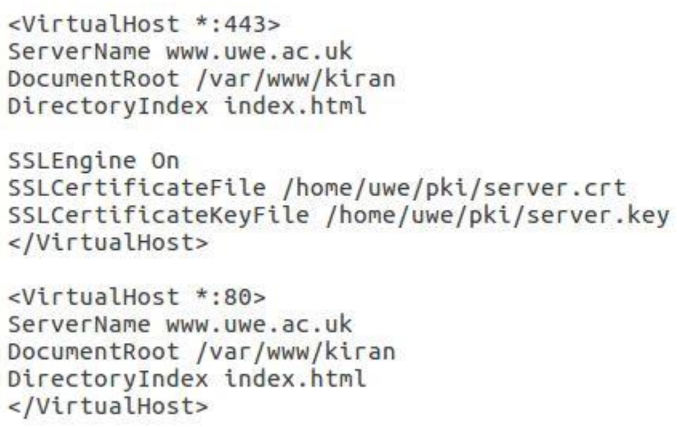
1.6 Task 6: Impersonating a Website (Man-in-the-Middle)

Now, we try to impersonate www.uwe.ac.uk. Firstly, we configure our DNS by adding an entry to the /etc/hosts file using the UWE url (Figure 19). Similar to previous task, we UWE's VirtualHost entry to the apache configuration file. Figure 20 shows this the added entry.



```
127.0.0.1       User
127.0.0.1       Attacker
127.0.0.1       Server
127.0.0.1       www.SeedLabSQLInjection.com
127.0.0.1       www.xsslabelgg.com
127.0.0.1       www.csrflabelgg.com
127.0.0.1       www.csrflabattacker.com
127.0.0.1       www.repackagingattacklab.com
127.0.0.1       www.seedlabclickjacking.com
127.0.0.1       uwekiranmohanraj.com
127.0.0.1       www.uwe.ac.uk
```

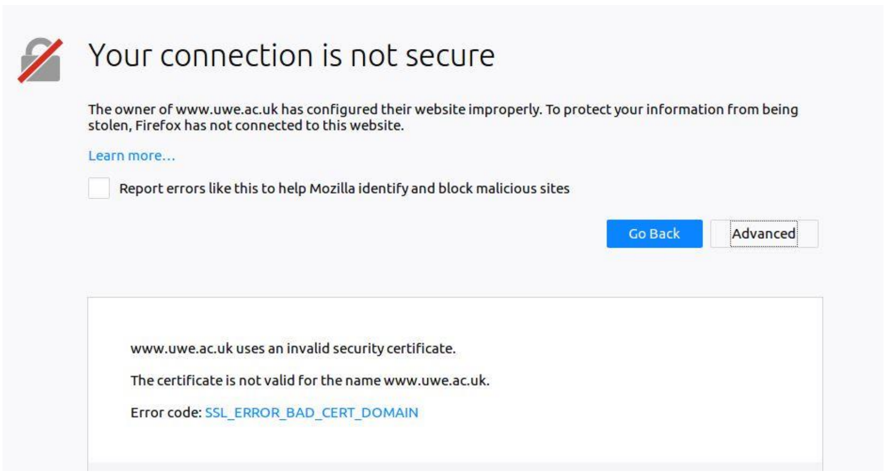
Figure 19 Configuring DNS for our hostname



```
<VirtualHost *:443>
ServerName www.uwe.ac.uk
DocumentRoot /var/www/kiran
DirectoryIndex index.html

SSLEngine On
SSLCertificateFile /home/uwe/pki/server.crt
SSLCertificateKeyFile /home/uwe/pki/server.key
</VirtualHost>

<VirtualHost *:80>
ServerName www.uwe.ac.uk
DocumentRoot /var/www/kiran
DirectoryIndex index.html
</VirtualHost>
```

Figure 20 VirtualHost entry

Now we browse our website https://www.uwe.ac.uk and see that we get an error saying that the connection is not secure. In more details option, we could see the common name of our UWE url and the certificate does not match, indicating the certificate is not meant for this domain.



Your connection is not secure

The owner of www.uwe.ac.uk has configured their website improperly. To protect your information from being stolen, Firefox has not connected to this website.

Learn more…

Report errors like this to help Mozilla identify and block malicious sites

www.uwe.ac.uk uses an invalid security certificate.

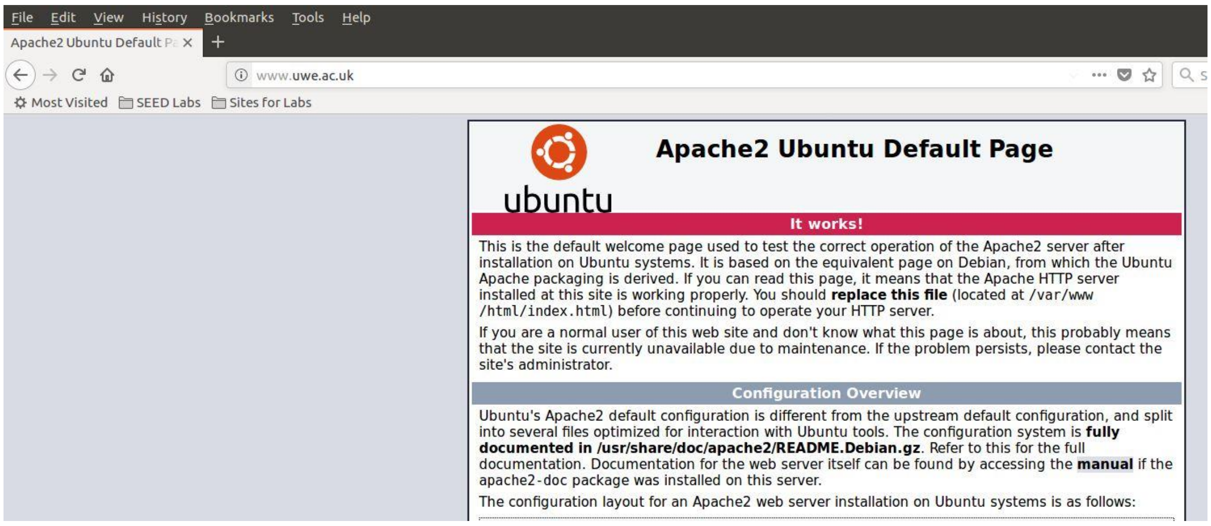
The certificate is not valid for the name www.uwe.ac.uk.

Error code: SSL_ERROR_BAD_CERT_DOMAIN

Figure 21 Browsing https://www.uwe.ac.uk

This proves that the MITM attack is defeated in the use of public key infrastructure. If the common name matched, then the website would have loaded. However, the common name will match only if the server had a valid certificate from the CA for its domain, which it didn't, in this case.

When we try to browse http://www.uwe.ac.uk, the we could see that the page has been redirected to Apache2 Default page (as shown in Figure 22). Thus, proving MITM attack can be implemented.



Apache2 Ubuntu Default Page

It works!

This is the default welcome page used to test the correct operation of the Apache2 server after installation on Ubuntu systems. It is based on the equivalent page on Debian, from which the Ubuntu Apache packaging is derived. If you can read this page, it means that the Apache HTTP server installed at this site is working properly. You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

If you are a normal user of this web site and don't know what this page is about, this probably means that the site is currently unavailable due to maintenance. If the problem persists, please contact the site's administrator.

Configuration Overview

Ubuntu's Apache2 default configuration is different from the upstream default configuration, and split into several files optimized for interaction with Ubuntu tools. The configuration system is **fully documented in /usr/share/doc/apache2/README.Debian.gz**. Refer to this for the full documentation. Documentation for the web server itself can be found by accessing the **manual** if the apache2-doc package was installed on this server.

The configuration layout for an Apache2 web server installation on Ubuntu systems is as follows:

Figure 22 Browsing http:www.uwe.ac.uk

1.7 Task 7: Alternative Approaches to (Certificate-Based) PKI (Research Task)

INTRODUCTION

Public-key infrastructure (PKI) is the basis for using public-key cryptography through key management. PKI helps in authenticating the public key of the user. If the authenticity is compromised, any user can use it to decrypt the messages, leading to problem of repudiation. In certificate-based PKI, a Certification Authority (CA) issues a digital certificate which authenticates the public key of the user. This digital certificate contains the public key and identity of the user. If the user wants to use other entities public key, they must verify its certificate for validating the authenticity of the public key. The major drawback of certificate-based PKI is scalability and certificate management. (Singh *et al.*, 2019).

HYBRID PKI SCHEMES

To overcome the drawbacks of certificate-based PKI, alternate approaches were proposed. Identity-based PKI (ID-PKI) was proposed by (Shamir, 1985), which used publicly known identity (eg. E-mail address) as public-key and the private key is sent to the owner of the identity (through Email). Although ID-PKI overcome the problem of distributing public key through certificates, it lacks scalability and support for identity management overhead (Singh *et al.*, 2019). To overcome the challenges of the certificate-based PKI and IDPKC, the hybrid approach was introduced which uses the positive features of both these models. In this report, we will be looking three hybrid PKI approaches:

A) Self- Certificate Scheme:

Lee (Lee and Kim, 2000) proposed this self-certificate scheme which address the issues of explicit authentication of self-certified key. This self-certificate is user generated for the public key by signing it with private key to the public key. The key pairs of the users are renewed by themselves. The authenticity of the Certificate Authority's certification is maintained without the user interacting with the CA for renewing the key pairs. The certificate-based scheme's revocation method is used by the Certificate Authority to revoke an issued key. The major advantage of this self-certificate scheme is that the user can renew their key pairs without interacting with the Certificate authority, meanwhile having the CA's authentication. The drawback of this scheme is that it has self-certificate overhead similar to the certificate-based PKI overhead (Lee and Kim, 2000).

B) Certificate based encryption (CBE) Scheme

Gentry (Gentry, 2003) introduced CBE scheme to provide simple revocation in certificate-based PKI. By exploiting the pairing, this scheme eases out the certificate management of the PKI. This scheme implemented the implicit certification from the IBE (identity-based encryption) no escrow from public-key certification, thus producing a hybrid PKI scheme. In this model, the user generates their own private key or public key pair and request a certificate from the Certificate authority. For decryption key and proof of certification, the Certificate authority generates implicit certificate by using IBE scheme. This helps in eliminating the third-party queries on the certificate status (Gentry, 2003).

Shao (Shao, 2011) propose a new Certificate-Based Encryption scheme from pairings provably secure in the chosen-key model. This scheme is derived from the CBE scheme of Gentry (Gentry, 2003), after making some improvements. Each client is allowed to choose its partial private key and partial public key independently. To deal with the certificate revocation problem, this paper uses the concept of the time period proposed by Gentry, which is added into certificate information to eliminate the need of the current certification status. The full public key of the client is just its certificate information with a time period. As results, anyone can encrypt messages under the certificate information and the time period so that the receiver can decrypt only if the receiver's partial public key is currently certified. However, if a client entirely trusts the authenticity of a receiver's partial public key, he could use the partial public key of the receiver directly to encrypt messages. Furthermore, this high security level is just at the cost of one more hashing and one more point scalar for encryption, which can be precomputed once for all. This scheme not only overcomes the weakness in the certified-key model of the Gentry scheme, but also raises efficiency.

C) Unified Public Key Infrastructure (UPKI) Scheme

Lee (Lee, 2010) introduces a new concept called unified public key infrastructure (UPKI) in which both certificate-based and ID-based cryptography are provided to users in a highly combined manner. This scheme assumes the existence of a trusted authority called key generation and certification authority (KGCA) who has the role of both CA and KGC. It checks identification information of user and issues a certificate for a user-chosen public key X. It also issues ID-based partial private key to the user. This paper assumes the existence of multiple KPAs (Key Privacy Agents) who provide key privacy service. In the proposed private key issuing protocol user is authenticated with certificate and user's certified public key X is used to blind the protocol messages such that only the legitimate user can retrieve the ID-based private key.

CONCLUSION

The demand for successfully implemented PKI systems is increasing as many organisations are looking to secure their communications. This report provides various hybrid PKI schemes involving certificates. There is need for some kind of certification is required in order to solve the problem of explicit authentication and scalability related issues. UPKI achieves all these problems except with some communication and complexity overhead.

1.8 References

Gentry, C. (2003) ‘Certificate-Based Encryption and the Certificate Revocation Problem’, in Biham, E. (ed.) *Advances in Cryptology --- EUROCRYPT 2003*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 272–293.

Lee, B. (2010) ‘Unified Public Key Infrastructure Supporting Both Certificate-Based and ID-Based Cryptography’, in *2010 International Conference on Availability, Reliability and Security*, pp. 54–61. doi: 10.1109/ARES.2010.49.

Lee, B. and Kim, K. (2000) ‘Self-certificate: PKI using self-certified key’, *Proc. of Conference on Information Security and Cryptology*, 10(1), pp. 65–73. Available at: http://cris.joongbu.ac.kr/publication/selfcert-CISC2000.pdf.

Shamir, A. (1985) ‘Identity-Based Cryptosystems and Signature Schemes’, in Blakley, G. R. and Chaum, D. (eds) *Advances in Cryptology*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 47–53.

Shao, Z. (2011) ‘Enhanced Certificate-Based Encryption from pairings’, *Computers & Electrical Engineering*, 37(2), pp. 136–146. doi: https://doi.org/10.1016/j.compeleceng.2011.01.007.

Singh, P. *et al.* (2019) ‘Towards a Hybrid Public Key Infrastructure (PKI): A Review’, *Cryptology ePrint Archive*, 509(Report 2019/784), pp. 1–19. Available at: https://eprint.iacr.org/2019/784.