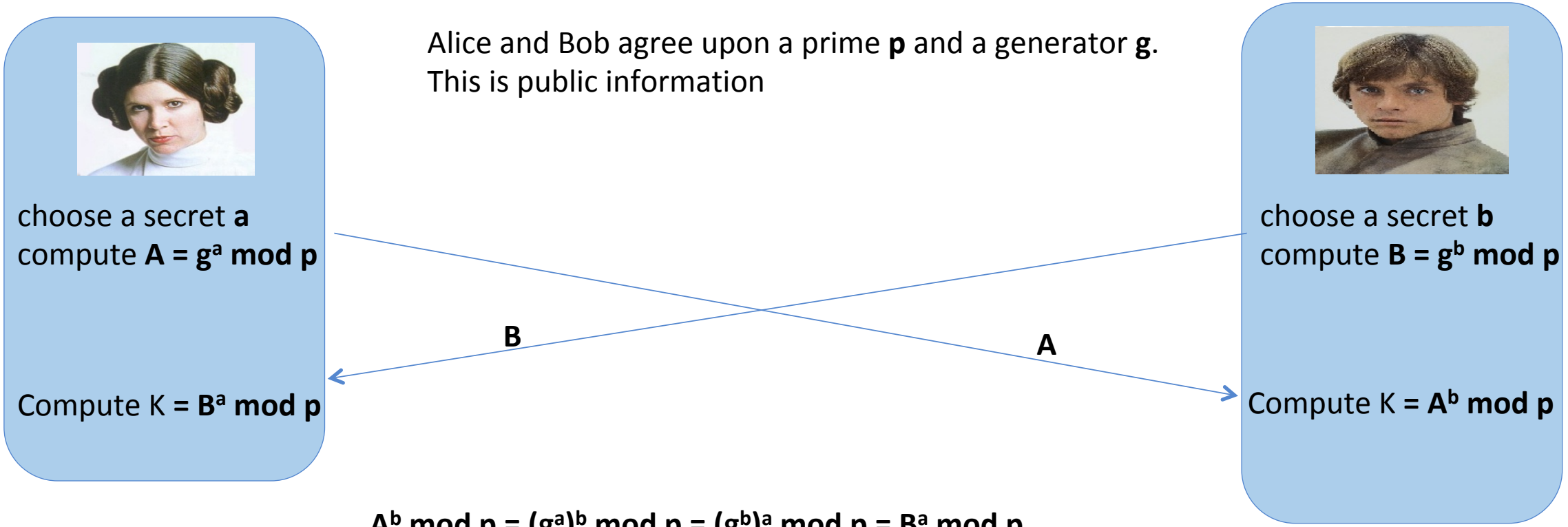
Public Key Infrastructure

Chester Rebeiro

IIT Madras

Recollect Diffie-Hellman Key Exchange

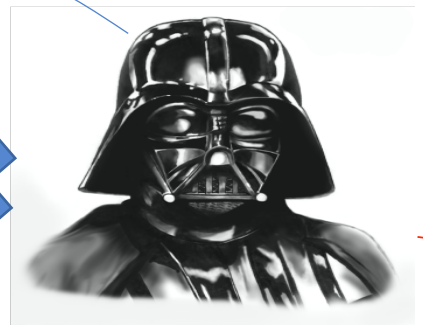
- **Key Establishment :** "Alice and Bob want to use a block cipher for encryption. How do they agree upon the secret key"



$$A^b \bmod p = (g^a)^b \bmod p = (g^b)^a \bmod p = B^a \bmod p$$

Man in the Middle Attack

Man in the middle
Intercepts messages



Alice
sends his public key

Sally
sends her public key

Alice decrypts
with her private
key

Sally decrypts
with her private
key and re-encrypts
With Alice's
public key

Bob encrypts
with Sally's public key

Man in the Middle Attack

Man in the middle
Intercepts messages

Alice
sends his public key

Sally
sends her

Alice
with h
key

public key

with Mallory

Fundamental Problem: Who is Alice?

Bob has no way to tell whether the public key he received belongs to Alice or not.

Certifying Authority

choose a secret **a**
compute **$A = g^a \bmod p$**

Compute K = **$B^a \bmod p$**

Digitally certificate
Public key of Bob (B)

Digitally certificate
Public key of Alice (A)

choose a secret **b**
compute **$B = g^b \bmod p$**

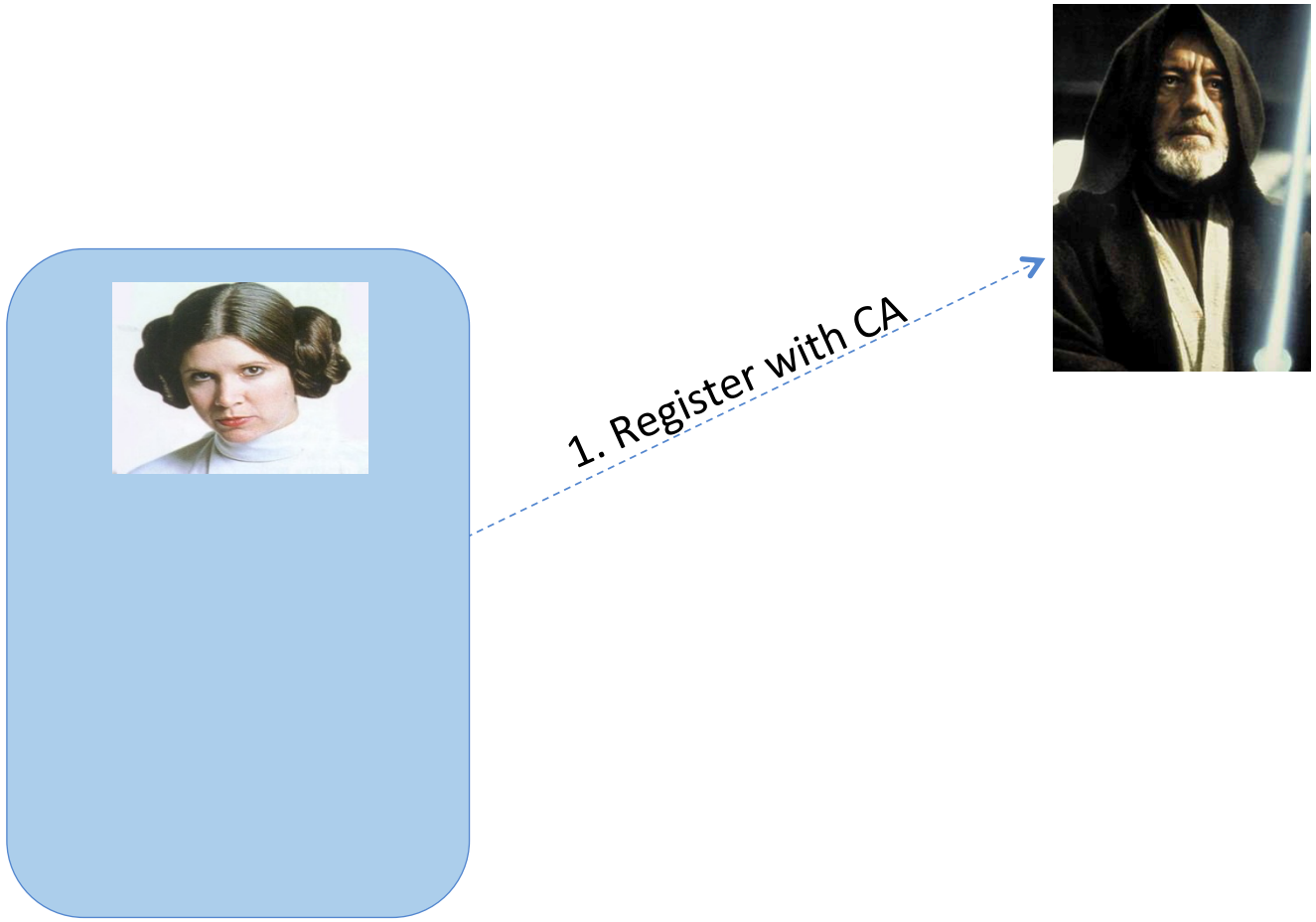
Compute K = **$A^b \bmod p$**

X.509 Digital Certificates

Contains

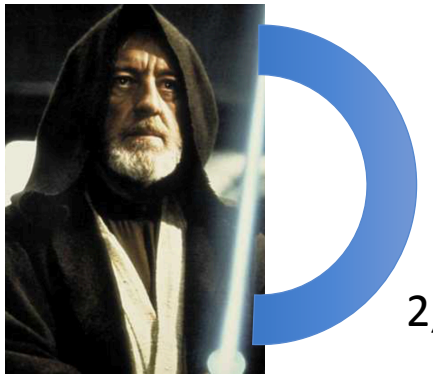
- Serial Number
- Issuer → *the certifying authority details*
- Subject → *information about the owner (who own's the public key for example Alice)*
- Public Key → *Alice's public key*
- Validity
- Signature → *The signature of the certificate signed by the certifying authority*

A more practical Perspective



A more practical Perspective

1. Register with CA

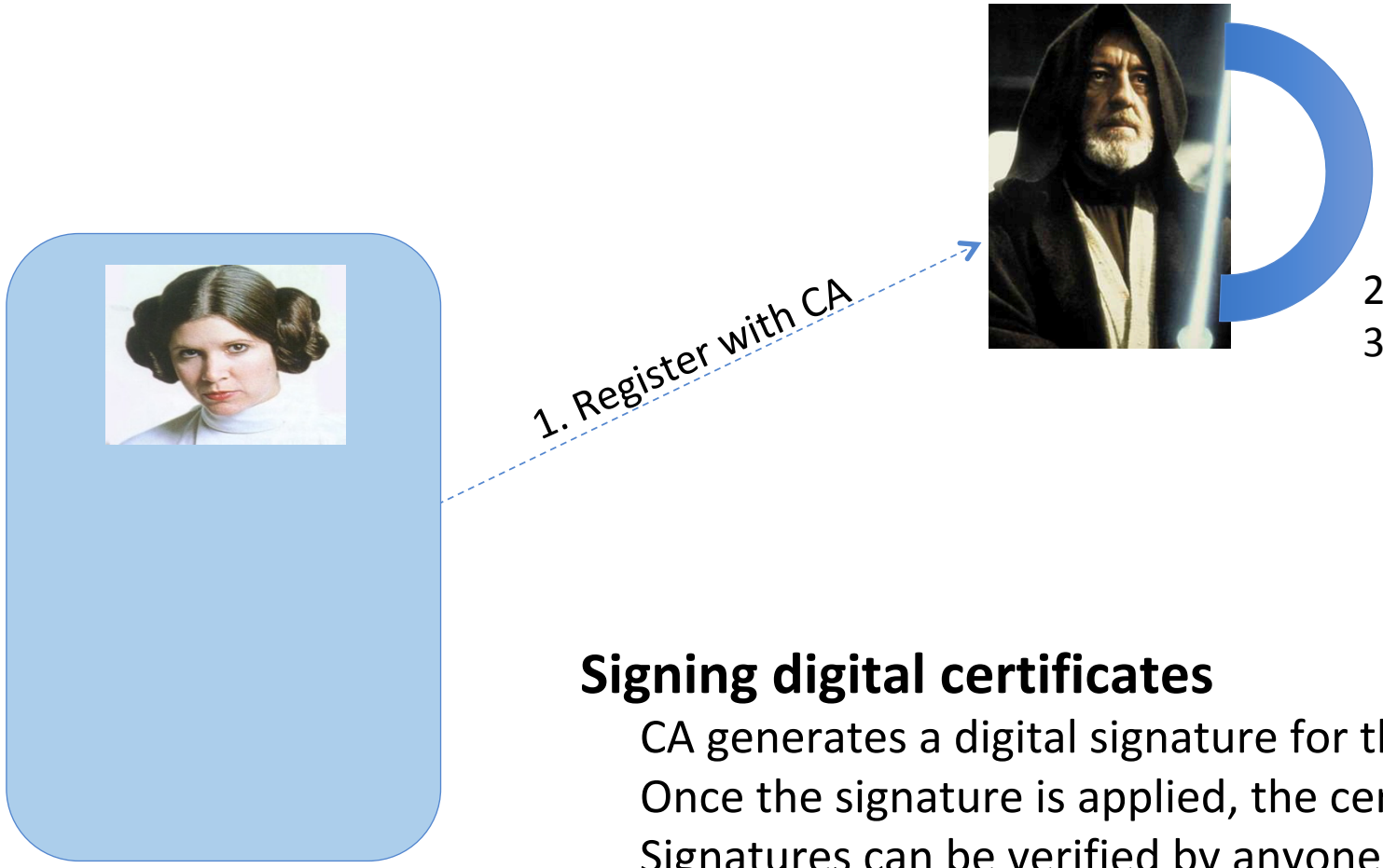


2, Verify Identity of Alice

Verify the subject

Ensure that the person applying for the certificate either owns or represents the identity in the subject field.

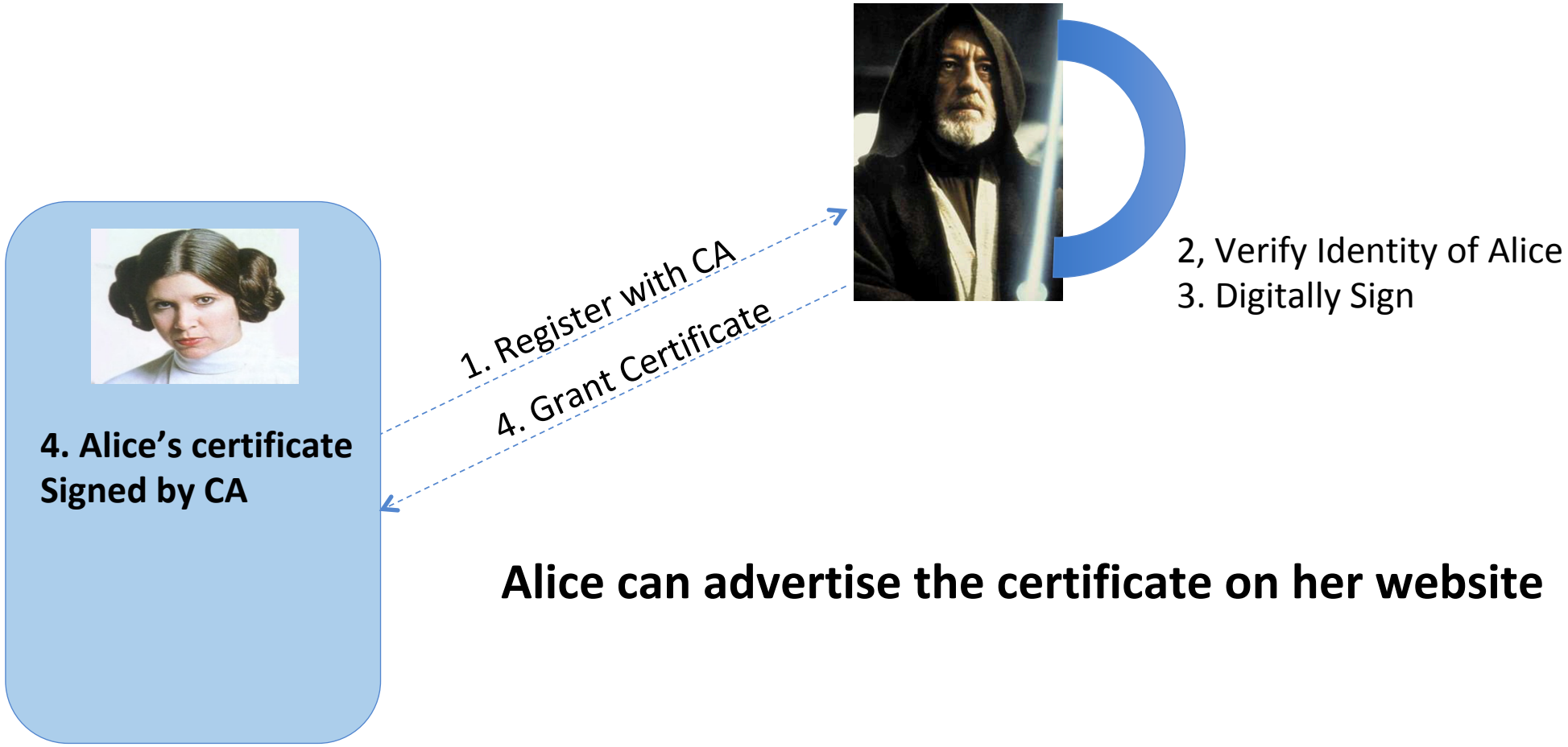
A more practical Perspective



Signing digital certificates

CA generates a digital signature for the certificate using its private key.
Once the signature is applied, the certificate cannot be modified.
Signatures can be verified by anyone with the CA's public key.

A more practical Perspective



Alice can advertise the certificate on her website

A more practical Perspective

Alice's certificate
Signed by CA

choose a secret **a**
compute **$A = g^a \bmod p$**

Compute K = **$B^a \bmod p$**

Alice's certificate

Bob's certificate

Bob'scertificate
Signed by CA

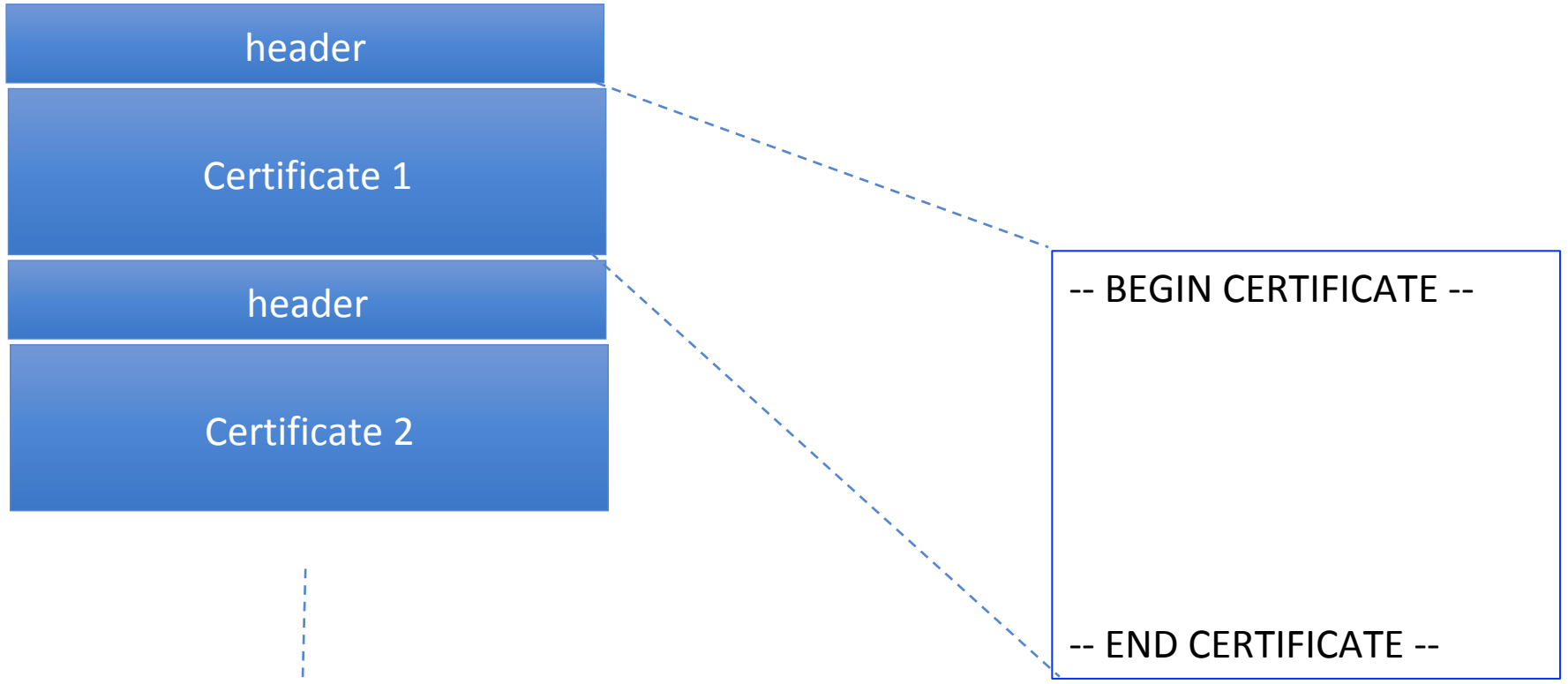
choose a secret **b**
compute **$B = g^b \bmod p$**

Compute K = **$A^b \bmod p$**

Fetching certificates with openssl

```
bellatrix:tmp chester$ openssl s_client -showcerts -connect www.paypal.com:443
```

Hostname : port



Fetching certificates with openssl

```
bellatrix:tmp chester$ openssl s_client -showcerts -connect www.paypal.com:443
```

Hostname : port

```
-----BEGIN CERTIFICATE-----
MIIH2DCCBsCgAwIBAgIQAVvaZl/ES3UXtogsHqvU3DANBgkqhkiG9w0BAQsFADB1
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGlnaWNlcnQuY29tMTQwMgYDVQQDEytEaWdpQ2VydCBTSEEyIEV4dGVuZGVk
IFZhbGlkYXRpb24gU2VydmVyIENBMB4XDTE4MDgxNDAwMDAwMFoXDTIwMDgxODEy
MDAwMFowgdwxHTAbBgNVBA8MFFByaXZhdGUgT3JnYW5pemF0aW9uMRMwEQYLKwYB
BAGCNzwCAQMTAlVTMRkwFwYLKwYBBAGCNzwCAQITCERlbGF3YXJlMRAwDgYDVQQF
EwczMDE0MjY3MQswCQYDVQQGEwJVUzETMBEGA1UECBMKQ2FsaWZvcm5pYTERMA8G
A1UEBxMIU2FuIEpvc2UxFTATBgNVBAoTDFBheVBhbCwgSW5jLjEUMBIGA1UECxML
```

```
kntHOFeVCqtS6BqQlmQ2jh7RhQAAAWU6rkRoAAAEAwBHMEUCIQDL83CcOZaDn1Zw
zRRK+PnOMv7rANqVOTO74RzQ+vNLRwIgEqj3eylzWjUybASfSHwTeid18BiY8XHE
cvXyXPflIGIwDQYJKoZIhvcNAQELBQADggEBAKHrnn/HFy4oL00LOJW7W8qeFDiM
7KYjJh87agfeTktBEf7u/feUjtAtJkI78j3oqbUOWIfvzlQqleqdmaQZknPpp+bc
4z9y2lcaMo8IIY2tN5jQVz5nZBCBvle/Dr+YiJur8Rh9cUox7LFfIl6VqN4CfsR9
5K7WNQLZINuNyYMDQiN8YKdNVTCwJryL706piCnhHfPFJHOpB3GbBI8cLTYr1sdp
5dXMg7vQdcCStA+TLiAV4GxSpqlIVpRF0YpqYbzjTiRne9ak/eG0//m4atvKBpXh
9ZXk76n7dH4/nv2u3h8dbt32AMTVozQCJiMaRlMlMElaNvcPwmGHNnEuvcs=
-----END CERTIFICATE-----
```

Cut and paste in a file paypal.pem

(PEM: privacy enhanced mail)

To view text equivalent of this, use

openssl x509 –in paypal.pem –text -noout

Example of X.509 Certificate (1st Part)

The CA's identity (Symantec)

The owner of the certificate (paypal)

```
Certificate:
Data:
  Serial Number:
          2c:d1:95:10:54:37:d0:de:4a:39:20:05:6a:f6:c2:7f
  Signature Algorithm: sha256WithRSAEncryption
  Issuer: C=US, O=Symantec Corporation, OU=Symantec Trust Network,
         CN=Symantec Class 3 EV SSL CA - G3
  Validity
     Not Before: Feb  2 00:00:00 2016 GMT
     Not After : Oct 30 23:59:59 2017 GMT
  Subject: 1.3.6.1.4.1.311.60.2.1.3=US/
          1.3.6.1.4.1.311.60.2.1.2=Delaware/
          businessCategory=Private Organization/
          serialNumber=3014267, C=US/
          postalCode=95131-2021, ST=California,
          L=San Jose/street=2211 N 1st St,
          O=PayPal, Inc., OU=CDN Support, CN=www.paypal.com
```

Example of X.509 Certificate (2nd Part)

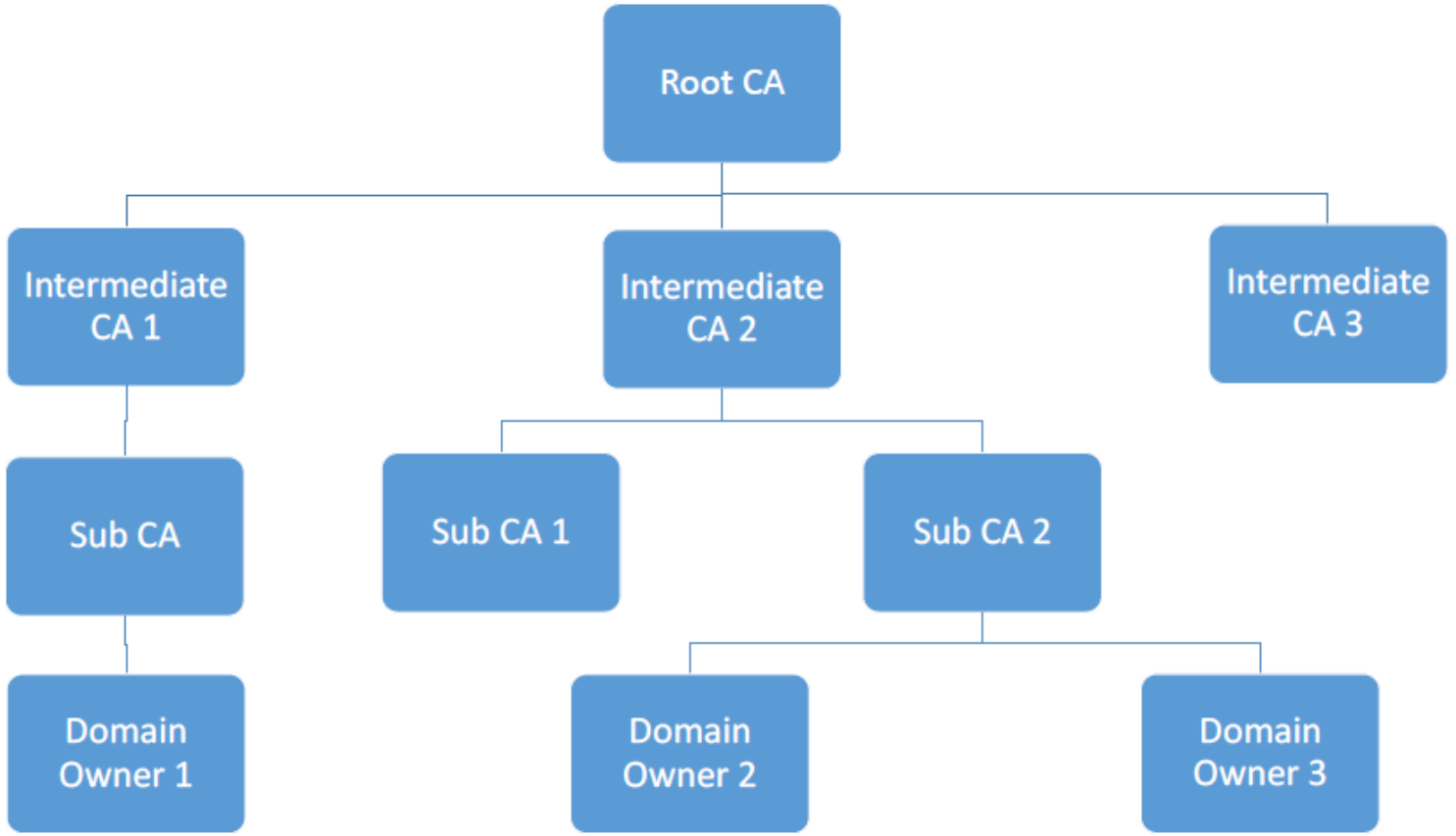
Public key

CA's signature

```
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
        Public-Key: (2048 bit)
        Modulus:
            00:da:43:c8:b3:a6:33:5d:83:c0:63:14:47:fd:6b:22:bd:
            bf:4e:a7:43:11:55:eb:20:8b:e4:61:13:ee:de:fe:c6:e2:
            ... (omitted) ...
            7a:15:00:c5:01:69:b5:10:16:a5:85:f8:fd:07:84:9a:c9:
        Exponent: 65537 (0x10001)
Signature Algorithm: sha256WithRSAEncryption
4b:a9:64:20:cc:77:0b:30:ab:69:50:d3:7f:de:dc:7c:e2:fb:93:84:fd:
78:a7:06:e8:14:03:99:c0:e4:4a:ef:c3:5d:15:2a:81:a1:b9:ff:dc:3a:
... (omitted) ...
fb:00:3e:7d:6a:de:cb:9f:ff:ef:8c:65:35:e4:22:b5:88:b2:48:32:1e:
```

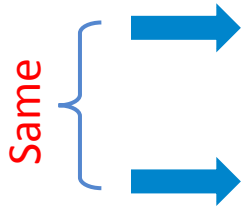
Who Certifies the CA?

There are many CAs in the real world, and they are organized in a hierarchical structure.



Root CAs and Self-Signed Certificate

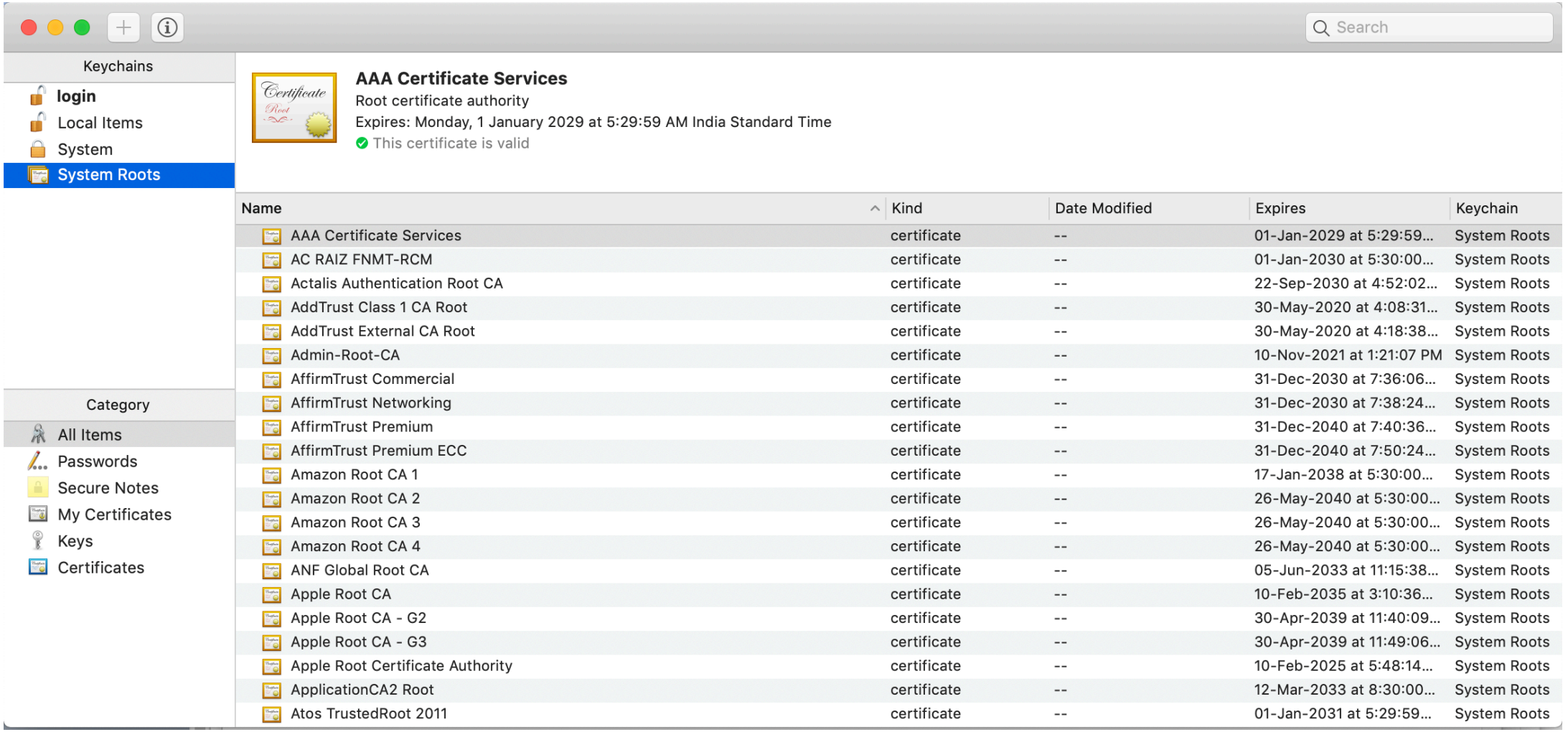
- A root CA's public key is also stored in an X.509 certificate. It is self-signed.
- Self-signed: the entries for the issuer and the subject are identical.




```
Issuer: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network,
    OU=(c) 2006 VeriSign, Inc. - For authorized use only,
    CN=VeriSign Class 3 Public Primary Certification Authority - G5
Subject: C=US, O=VeriSign, Inc., OU=VeriSign Trust Network,
    OU=(c) 2006 VeriSign, Inc. - For authorized use only,
    CN=VeriSign Class 3 Public Primary Certification Authority - G5
```

- How can they be trusted?
  - Public keys of root CAs are pre-installed in the OS, browsers and other software

Root CAs in Mac OS



Intermediate CAs and Chain of Trust

```
$ openssl s_client -showcerts -connect www.paypal.com:443

Certificate chain
 0 s: ... /CN=www.paypal.com
   i: ... /CN=Symantec Class 3 EV SSL CA - G3
-----BEGIN CERTIFICATE-----
MIIHWTCCBkGgAwIBAgIQLNGVEFQ30N5KOSAFavbCfzANBgkqhkiG9w0BAQsFADB3
...
-----END CERTIFICATE-----
 1 s: ... /CN=Symantec Class 3 EV SSL CA - G3
   i: ... /CN=VeriSign Class 3 Public Primary Certification
              Authority - G5
-----BEGIN CERTIFICATE-----
MIIFKzCCBBOgAwIBAgIQfuFKb2/v8tN/P61lTTratDANBgkqhkiG9w0BAQsFADCB
...
-----END CERTIFICATE-----
```

Paypal's certificate

Intermediate CA's certificate

B

A

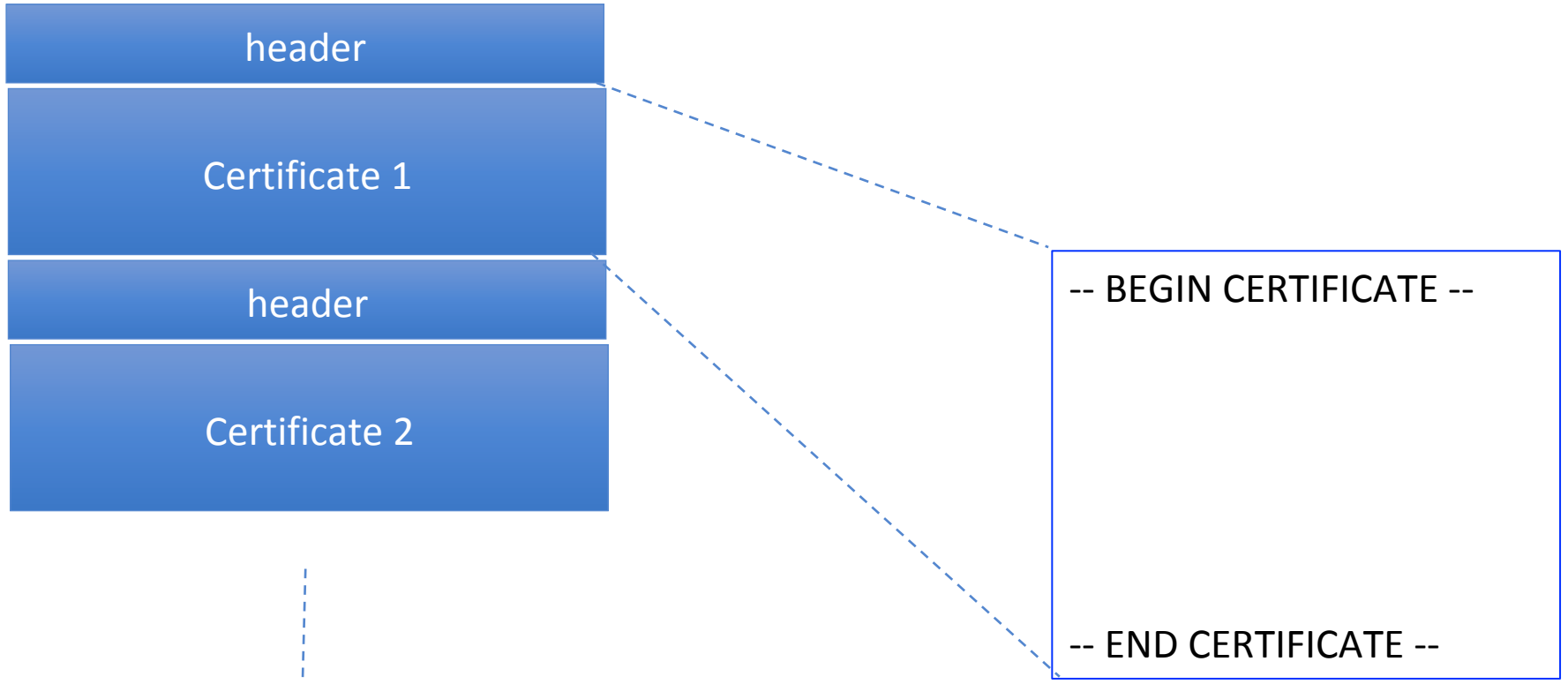
A is used to verify B

Something else is need to verify A (certificate from another intermediate CA or root CA)

Fetching certificates with openssl

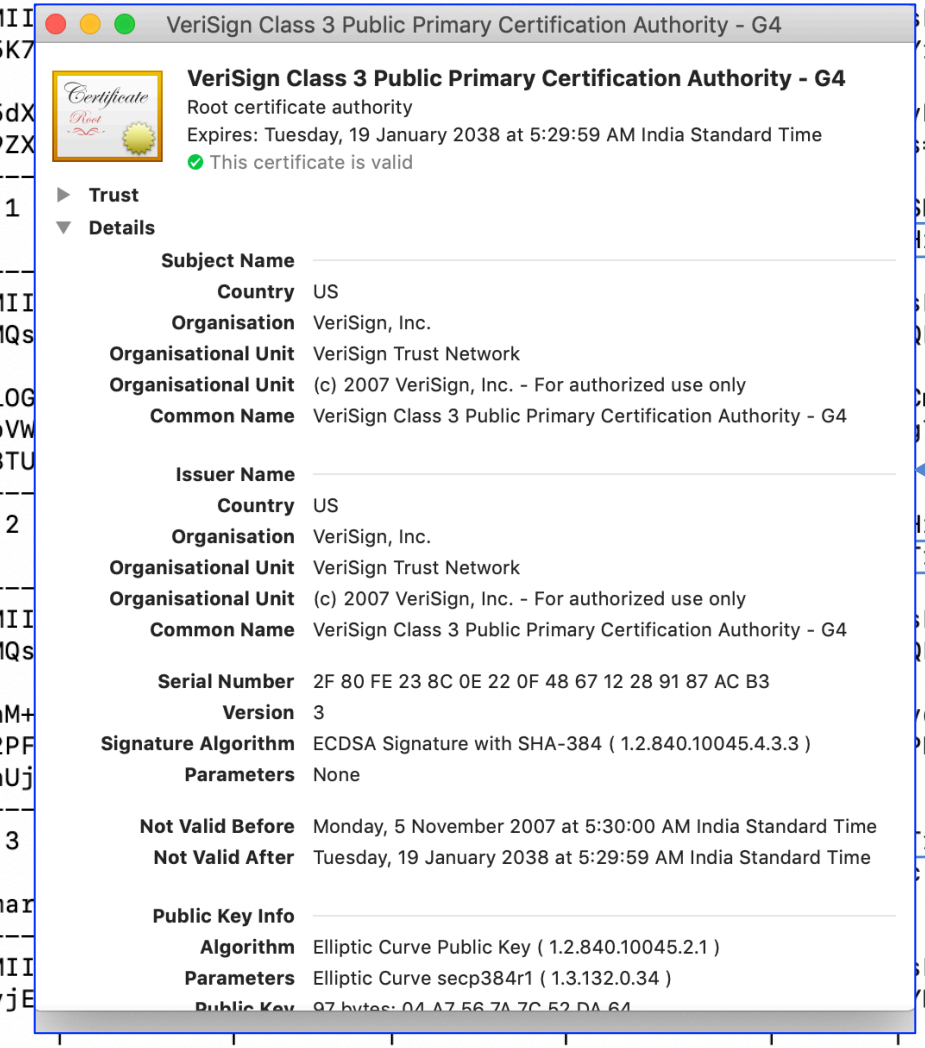
```
bellatrix:tmp chester$ openssl s_client -showcerts -connect www.paypal.com:443
```

Hostname : port



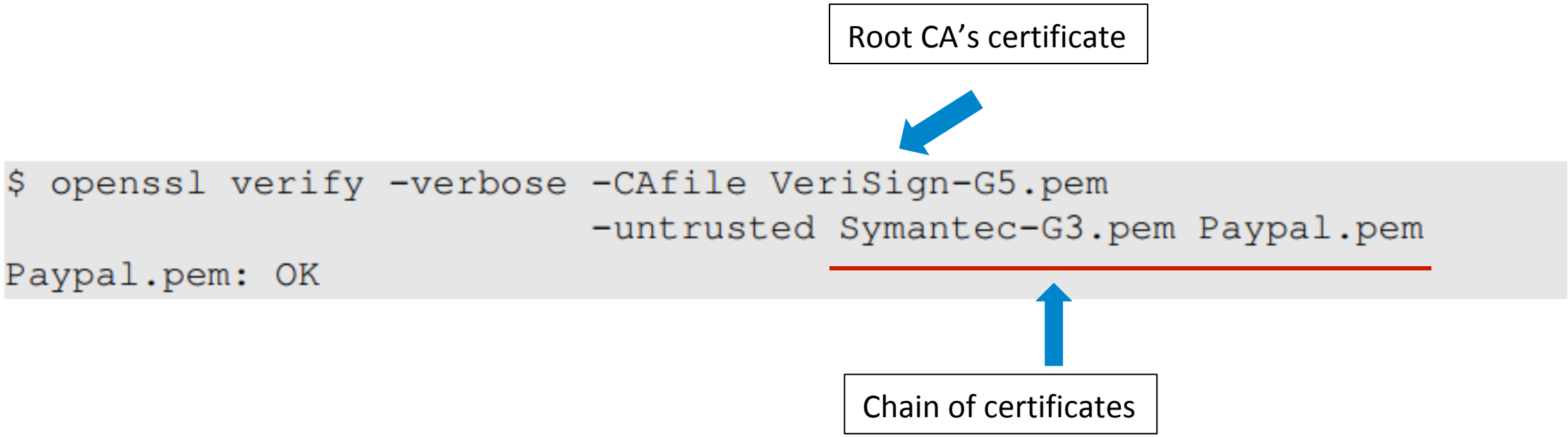
```
Certificate chain
 0 s:/businessCategory=Private Organization/jurisdictionCountryName=US/jurisdictionStateOrProvinceName=Delaware/serialNumber=3014267/C=US
/ST=California/L=San Jose/O=PayPal, Inc./OU=CDN Support/CN=www.paypal.com
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 Extended Validation Server CA
-----BEGIN CERTIFICATE-----
MIIH2DCCBsCgAwIBAgIQAVvaZl/ES3UXtogsHqvU3DANBgkqhkiG9w0BAQsFADB1
5K7WNQLZINuNyYMDQiN8YKdNVTCwJryL706piCnhHfPFJHOpB3GbBI8cLTYr1sdp
    |          |         |          |             |       |
5dXMg7vQdcCStA+TLiAV4GxSpqlIVpRF0YpqYbzjTiRne9ak/eG0//m4atvKBpXh
9ZXk76n7dH4/nv2u3h8dbt32AMTVozQCJiMaRlMlMElaNvcPwmGHNnEuvcs=
-----END CERTIFICATE-----
 1 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 Extended Validation Server CA
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance EV Root CA
-----BEGIN CERTIFICATE-----
MIIEtjCCA56gAwIBAgIQDHmpRLCMEZUgkmFf4msdgzANBgkqhkiG9w0BAQsFADBs
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
    |          |         |          |             |       |
1OG9d4Q3A84ytciagRpKkk47RpqF/oOi+Z6Mo8wNXrM9zwR4jxQUezKcxwCmXMS1
oVWNWlZopCJwqjyBcdmdqEU79OX2olHdx3ti6G8MdOu42vi/hw15UJGQmxg7kVkn
8TUoE6smftX3eg==
-----END CERTIFICATE-----
 2 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance EV Root CA
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Transition RSA Root
-----BEGIN CERTIFICATE-----
MIIEgTCCA2mgAwIBAgIQDt+vRguxNkcljEV7K5Y1gDANBgkqhkiG9w0BAQsFADBm
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
    |          |         |          |             |       |
hM+Z1af6ar1CJt+VYK7U6hY9CdP+hLx1OmbkVv91aXdlZhrZHQrV6fobmKych0zd
2PFms2BYDT+lP4dX8MNvltleZ0EHvHPN+K49jdqX6JMi8VsCsUTLPB5e4tPB3rKB
hUjKXh8=
-----END CERTIFICATE-----
 3 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert Transition RSA Root
   i:/C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3 Public Pri
mary Certification Authority - G5
-----BEGIN CERTIFICATE-----
MIIE5DCCA8ygAwIBAgIQeK6kMcFc63V7DYphCnSOZzANBgkqhkiG9w0BAQsFADCB
yjELMAkGA1UEBhMCVVMxFzAVBgNVBAoTDlZlcmlTaWduLCBJbmMuMR8wHQYDVQQL
    |          |         |          |             |       |
XGXJQtObNTbifmDf5sMk49D8izYeWji81MeMOwc1ZCJG3maKNFtQxUKVaN0MhL4s
5OQuQgBg+R7XDT1ApvA7XZsXB7fyMEfkiwbXogY3KzyiqYLoDaPjG0zlkUP+PXi5
A4rg1sEFqCw=
-----END CERTIFICATE-----
```

```
Certificate chain
 0 s:/businessCategory=Private Organization/jurisdictionCountryName=US/jurisdictionStateOrProvinceName=Delaware/serialNumber=3014267/C=US
/ST=California/L=San Jose/O=PayPal, Inc./OU=CDN Support/CN=www.paypal.com
   i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 Extended Validation Server CA
-----BEGIN CERTIFICATE-----
MII                                                                    FADB1
5K7                                                                    r1sdp

5dX                                                                    KBpXh
9ZX                                                                    =
---
 1                                                                     HA2 Extended Validation Server CA
                                                                       igh Assurance EV Root CA
---
MII                                                                    FADBs
MQs                                                                    LExB3

1OG                                                                    mXMS1
oVW                                                                    g7kVkn
8TU
---
 2                                                                     igh Assurance EV Root CA
                                                                       ransition RSA Root
---
MII                                                                    FADBm
MQs                                                                    LExB3

hM+                                                                    ch0zd
2PF                                                                    B3rKB
hUj
---
 3                                                                     ransition RSA Root
                                                                       ) 2006 VeriSign, Inc. - For authorized use only/CN=VeriSign Class 3 Public Pri

mar
---
MII                                                                    FADCB
yjE                                                                    DVQQL

XGXJQtObNTbifmDf5sMk49D8izYeWji81MeMOwc1ZCJG3maKNFtQxUKVaN0MhL4s
5OQuQgBg+R7XDT1ApvA7XZsXB7fyMEfkiwbXogY3KzyiqYLoDaPjG0zlkUP+PXi5
A4rg1sEFqCw=
-----END CERTIFICATE-----
```

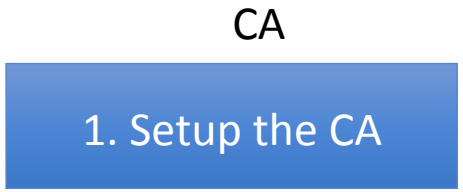


Manually Verifying a Certificate Chain

- Paypal.pem: Save Paypal's certificate to a file called
- Symatec-g3.pem: Save certificate from "Symantec Class 3 EV SSL CA – G3"
- VeriSign-G5.pem: Save the VeriSign-G5's certificate from the browser



The Entire Process



1. Setup the CA

```
[bellatrix:~ chester$ mkdir demoCA
[bellatrix:~ chester$ cd demoCA/
[bellatrix:demoCA chester$ mkdir certs crl mewcerts
[bellatrix:demoCA chester$ touch index.txt serial
[bellatrix:demoCA chester$ echo 1000 > serial
[bellatrix:demoCA chester$ openssl req -x509 -newkey rsa:4096 -sha256 -days 3650
-keyout modelCA_key.pem -out modelCA_cert.pem
Generating a 4096 bit RSA private key
...........++
.....................................................................................
........++
writing new private key to 'modelCA_key.pem'
[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) []:in
State or Province Name (full name) []:Tamil Nadu
Locality Name (eg, city) []:Chennai
Organization Name (eg, company) []:IIT Madras
Organizational Unit Name (eg, section) []:Computer Sc and Engg
Common Name (eg, fully qualified host name) []:cse.iitm.ac.in
Email Address []:chester@cse.iitm.ac.in
[bellatrix:demoCA chester$ ls
```

CA's self signed certificate

CA's public-private key (password protected)

1. Setup the CA

modelCA's certificate

```
bellatrix:demoCA chester$ openssl x509 -in modelCA_cert.pem -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 14292097484872891496 (0xc657baf0911a3c68)
    Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=in, ST=Tamil Nadu, L=Chennai, O=IIT Madras, OU=Computer Sc and Engg, CN=cse.
iitm.ac.in/emailAddress=chester@cse.iitm.ac.in
        Validity
            Not Before: Mar 19 14:16:54 2019 GMT
            Not After : Mar 16 14:16:54 2029 GMT
        Subject: C=in, ST=Tamil Nadu, L=Chennai, O=IIT Madras, OU=Computer Sc and Engg, CN=cse
.iitm.ac.in/emailAddress=chester@cse.iitm.ac.in
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:cf:61:62:39:46:4d:30:98:e1:5a:4c:19:29:0d:
                    24:fa:b3:03:40:00:0d:1f:8d:b2:e9:72:cd:be:da:
                          |          |          |        |
                    12:40:f5:9a:93:ed:48:a0:c5:6b:92:57:22:bf:04:
                    97:01:20:d9:7a:13:31:15:87:41:38:35:70:a5:2c:
                    73:7c:bd
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha256WithRSAEncryption
         3d:af:87:6b:dc:f6:b9:0d:a6:85:be:8a:63:b7:b9:b8:df:7b:
         58:4a:9a:40:b7:da:b7:9a:9c:bc:8e:74:2c:f8:e8:27:a6:da:
             |           |          |          |        |
         36:8b:95:2c:2e:84:f3:b8:43:b3:c8:13:68:61:dc:ab:ff:80:
         2f:15:01:c7:71:0e:4f:d3
```

Self signed

The Entire Process

CA

1. Setup the CA

user

1. Generate Keys

1. User Generate Keys

```
bellatrix:client chester$ openssl genrsa -aes128 -out bank_key.pem 2048
Generating RSA private key, 2048 bit long modulus
.....................................................................+++
.....................................................+++
e is 65537 (0x10001)
Enter pass phrase for bank_key.pem:
Verifying - Enter pass phrase for bank_key.pem:
```

```
bellatrix:demoCA chester$ openssl rsa -noout -text -in modelCA_key.pem
Enter pass phrase for modelCA_key.pem:
Private-Key: (4096 bit)
modulus:
    00:cf:61:62:39:46:4d:30:98:e1:5a:4c:19:29:0d:
    12:40:f5:9a:93:ed:48:a0:c5:6b:92:57:22:bf:04:
publicExponent: 65537 (0x10001)
privateExponent:
    21:9b:b9:ac:68:8d:47:eb:ee:d1:90:75:9f:66:8c:
    be:76:d1:4b:06:f7:0a:6c:84:7e:3d:b3:67:49:35:
prime1:
    00:f4:c9:7f:1a:c7:68:c5:01:d4:e3:7b:d3:fb:d8:
    f8:47:8b:f2:41:b7:90:b2:ac:b3:91:e5:b0:8c:af:
prime2:
    00:d8:e1:3c:88:ba:dd:02:1f:4e:52:82:e4:7f:1d:
    5e:cf:b7:33:e2:7d:6a:09:cd:4f:a1:99:58:71:dc:
exponent1:
    02:5b:5a:4c:f0:b4:92:89:04:fa:b7:bb:7f:c6:44:
    92:eb:87:81:b5:31:b2:1b:99:2c:61:73:4a:d9:3a:
exponent2:
    1b:c8:df:44:75:0c:13:55:87:67:32:b5:ab:43:45:
    f6:ed:75:8c:32:9c:ff:1c:7a:73:2d:7e:13:38:29:
coefficient:
    72:55:d8:83:3f:8f:dc:c5:ba:16:49:34:46:2d:c3:
    fa:3f:df:96:de:de:c9:33:dc:14:be:97:43:e1:e9:
```

n

Public key (A)

Private key (a)

p

q

ap

aq

q^{-1}

1. User Generate Keys

$n = p \times q$

The Entire Process

CA

1. Setup the CA

user

1. Generate Keys

2. Generate CSR
(certi signing req)

2. Generate CSR (certi signing req)

```
bellatrix:client chester$ openssl req -new -key bank_key.pem -out bank.csr -sha256
Enter pass phrase for bank_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) []:IN
State or Province Name (full name) []:Tamil Nadu
Locality Name (eg, city) []:Chennai
Organization Name (eg, company) []:IIT M
Organizational Unit Name (eg, section) []:Computer Sc and Engg
Common Name (eg, fully qualified host name) []:cse.iitm.ac.in
Email Address []:chester@cse.iitm.ac.in

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:cs6500
```

2. Generate CSR (certi signing req)

```
bellatrix:client chester$ openssl req -in bank.csr -text -noout
Certificate Request:
    Data:
        Version: 0 (0x0)
        Subject: C=IN, ST=Tamil Nadu, L=Chennai, O=IIT M, OU=Computer Sc and Engg, CN=cse.iitm
.ac.in/emailAddress=chester@cse.iitm.ac.in
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:bf:73:0c:66:3a:4b:49:00:a8:fc:2b:bf:d7:2c:
                    26:82:3e:5d:d2:1c:57:55:d0:6d:c7:1d:ec:85:c1:
                           |         |         |         |
                    4e:d5:a3:3f:64:f4:43:47:e0:1e:48:97:fa:f6:4f:
                Exponent: 65537 (0x10001)
        Attributes:
            challengePassword        :unable to print attribute
    Signature Algorithm: sha256WithRSAEncryption
         04:ea:38:df:d6:e2:4a:08:08:d0:3a:26:b9:4b:e8:7c:9f:e9:
         c7:5f:84:52:c0:7d:5d:bb:98:5d:f9:f6:c2:a6:15:79:ce:e1:
              |         |         |         |         |
         df:26:d0:9f:52:a1:09:0b:ac:6b:25:61:ca:29:a8:f7:55:4e:
         0b:b8:c4:db
```

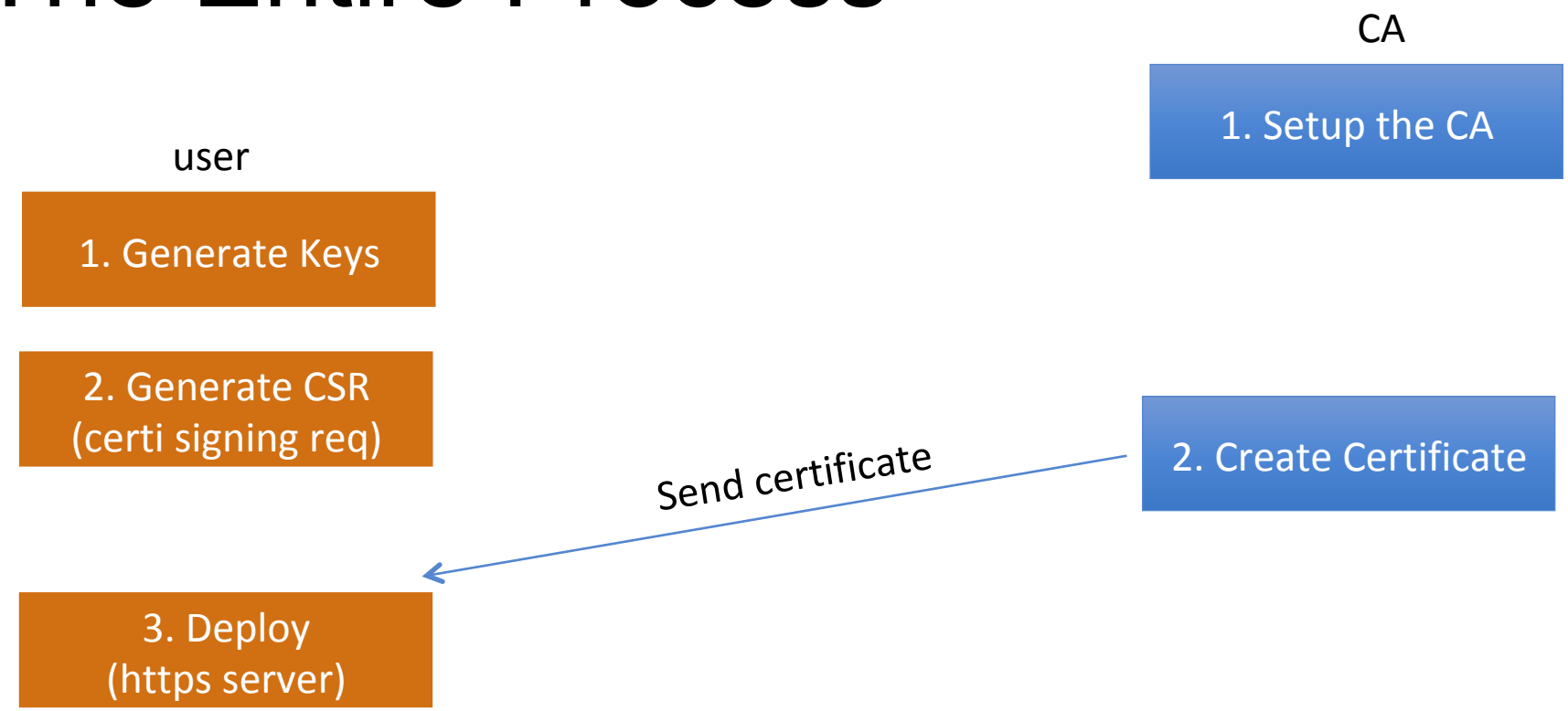
Signed with the bank's private key (self signed)

The Entire Process

2. Create Certificate

```
bellatrix:demoCA chester$ openssl ca -config ca.conf -in ../client/bank.csr -out bank_cert.pem
 -md sha256 -cert modelCA_cert.pem -keyfile modelCA_key.pem
Using configuration from ca.conf
Enter pass phrase for modelCA_key.pem:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'IN'
stateOrProvinceName   :ASN.1 12:'Tamil Nadu'
localityName          :ASN.1 12:'Chennai'
organizationName      :ASN.1 12:'IIT M'
organizationalUnitName:ASN.1 12:'Computer Sc and Engg'
commonName            :ASN.1 12:'cse.iitm.ac.in'
emailAddress          :IA5STRING:'chester@cse.iitm.ac.in'
Certificate is to be certified until Mar 16 16:16:52 2029 GMT (3650 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
```

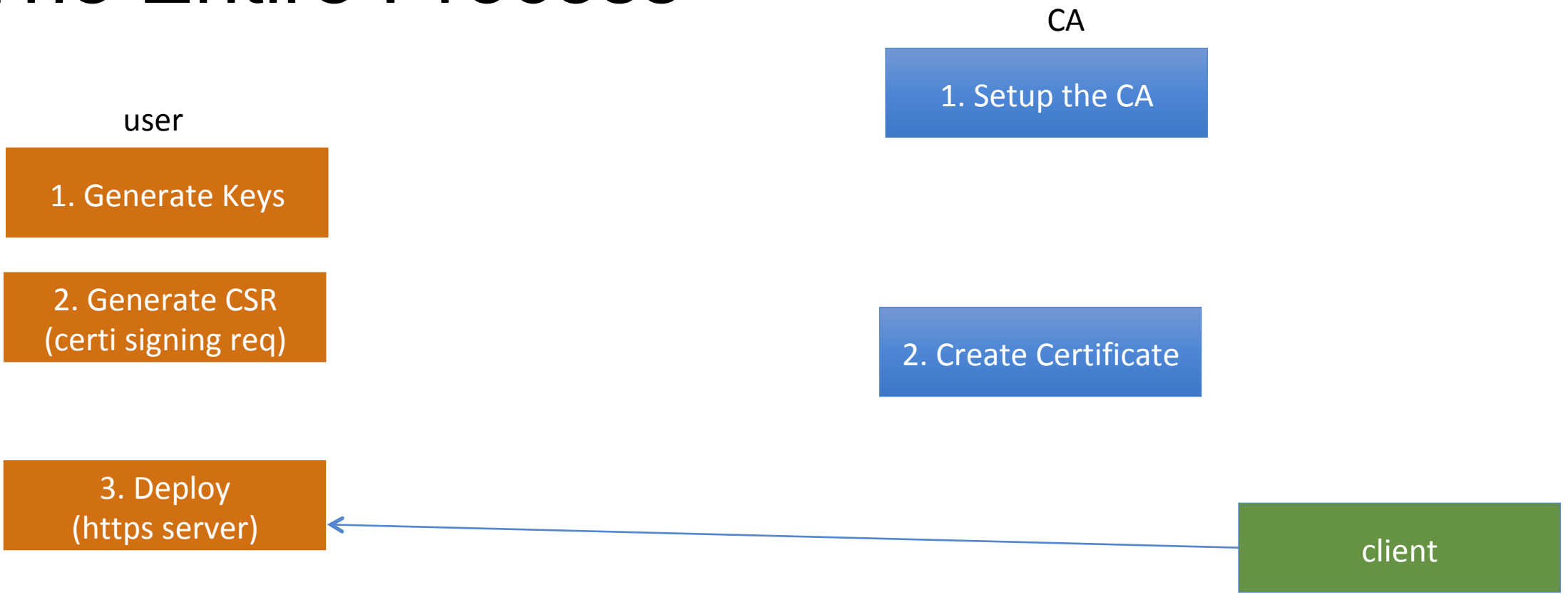
The Entire Process



3. Deploy

```
bellatrix:client chester$ openssl s_server -key bank_key.pem -cert bank_cert.pem -accept 44330
 -www
Enter pass phrase for bank_key.pem:
Using auto DH parameters
Using default temp ECDH parameters
ACCEPT
4709049964:error:140370E5:SSL routines:ACCEPT_SR_KEY_EXCH:ssl handshake failure:/BuildRoot/Lib
rary/Caches/com.apple.xbs/Sources/libressl/libressl-22.200.4/libressl-2.6/ssl/ssl_pkt.c:956:
ACCEPT
```

The Entire Process



A client fails to connect because it cannot verify the first (root) Certificate (modelCA)

client

```
bellatrix:demoCA chester$ openssl s_client -connect cse.iitm.ac.in:44330
CONNECTED(00000003)
depth=0 C = IN, ST = Tamil Nadu, O = IIT M, OU = Computer Sc and Engg, CN = cse.
iitm.ac.in
verify error:num=20:unable to get local issuer certificate
verify return:1
depth=0 C = IN, ST = Tamil Nadu, O = IIT M, OU = Computer Sc and Engg, CN = cse.
iitm.ac.in
verify error:num=21:unable to verify the first certificate
verify return:1
---
Certificate chain
 0 s:/C=IN/ST=Tamil Nadu/O=IIT M/OU=Computer Sc and Engg/CN=cse.iitm.ac.in
   i:/C=in/ST=Tamil Nadu/L=Chennai/O=IIT Madras/OU=Computer Sc and Engg/CN=cse.i
itm.ac.in/emailAddress=chester@cse.iitm.ac.in
---
```

```
    Start Time: 1553014493
    Timeout   : 7200 (sec)
    Verify return code: 21 (unable to verify the first certificate)
---
```

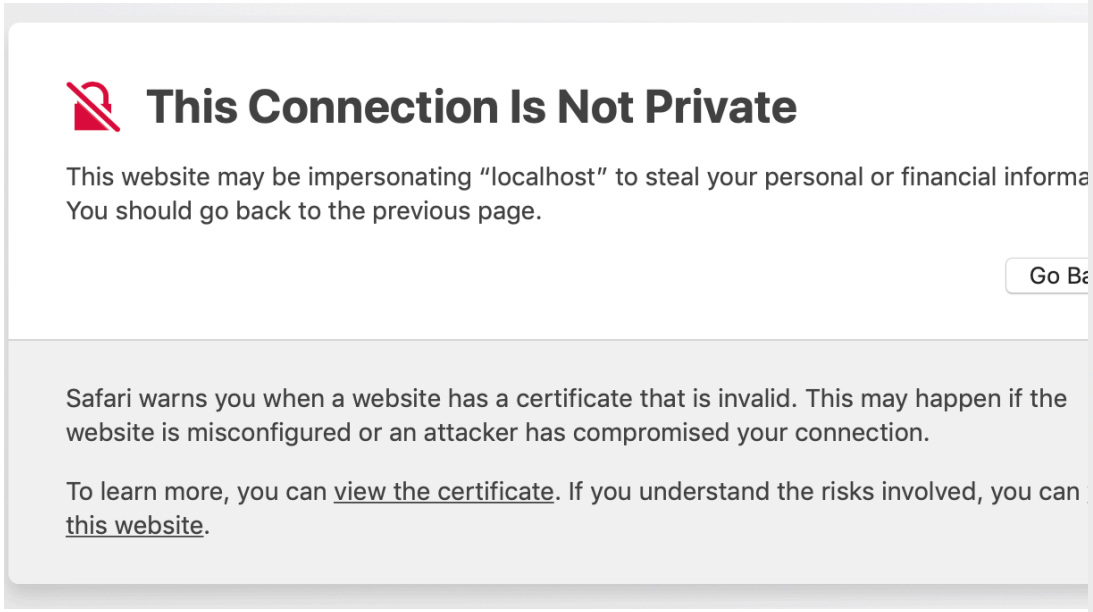
A client connects if the modelCAs certificate is known

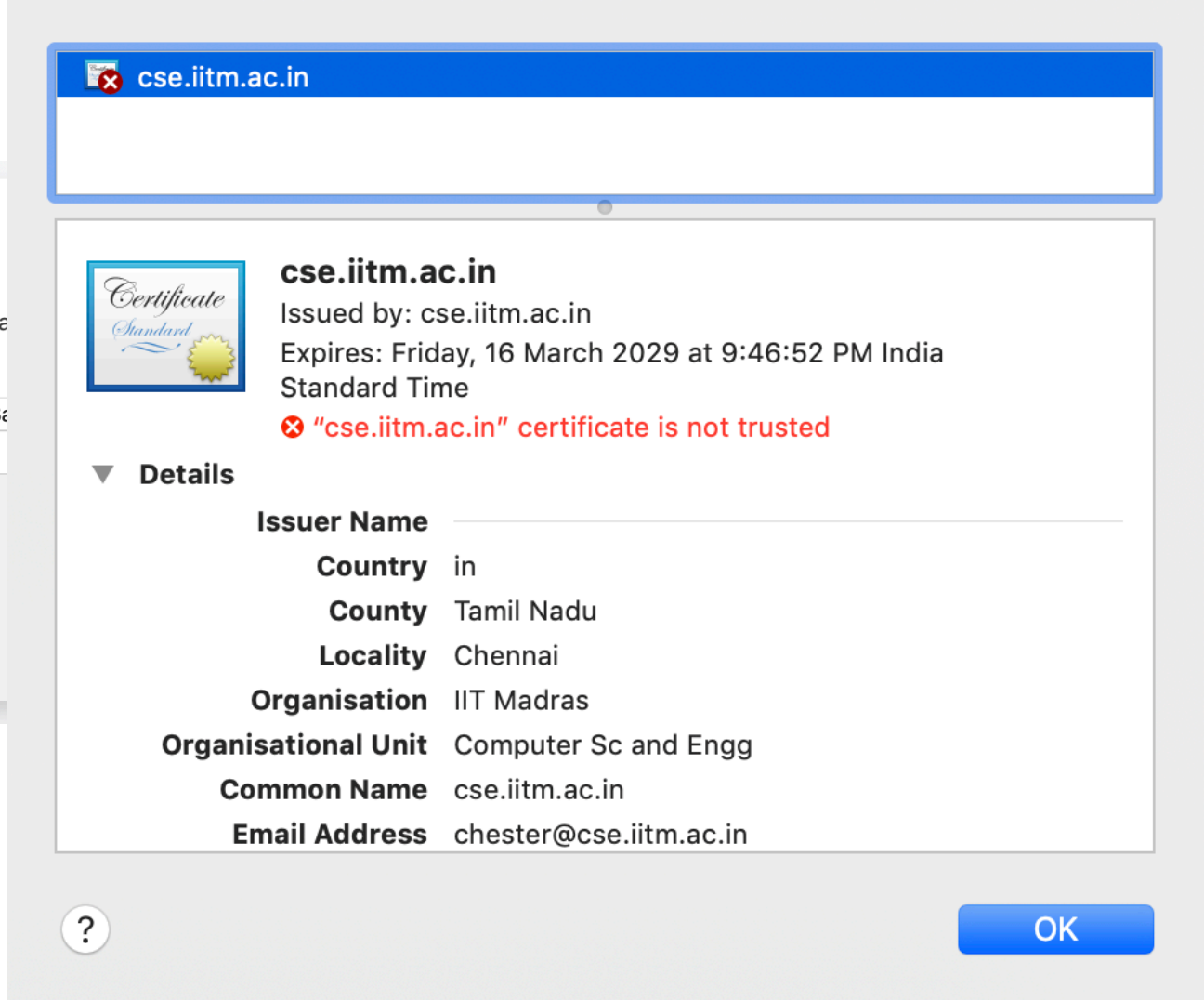
client

```
bellatrix:demoCA chester$ openssl s_client -connect cse.iitm.ac.in:44330 -CAfile
 ./modelCA_cert.pem
CONNECTED(00000003)
depth=1 C = in, ST = Tamil Nadu, L = Chennai, O = IIT Madras, OU = Computer Sc a
nd Engg, CN = cse.iitm.ac.in, emailAddress = chester@cse.iitm.ac.in
verify return:1
depth=0 C = IN, ST = Tamil Nadu, O = IIT M, OU = Computer Sc and Engg, CN = cse.
iitm.ac.in
verify return:1
---
Certificate chain
 0 s:/C=IN/ST=Tamil Nadu/O=IIT M/OU=Computer Sc and Engg/CN=cse.iitm.ac.in
   i:/C=in/ST=Tamil Nadu/L=Chennai/O=IIT Madras/OU=Computer Sc and Engg/CN=cse.i
itm.ac.in/emailAddress=chester@cse.iitm.ac.in
---
```

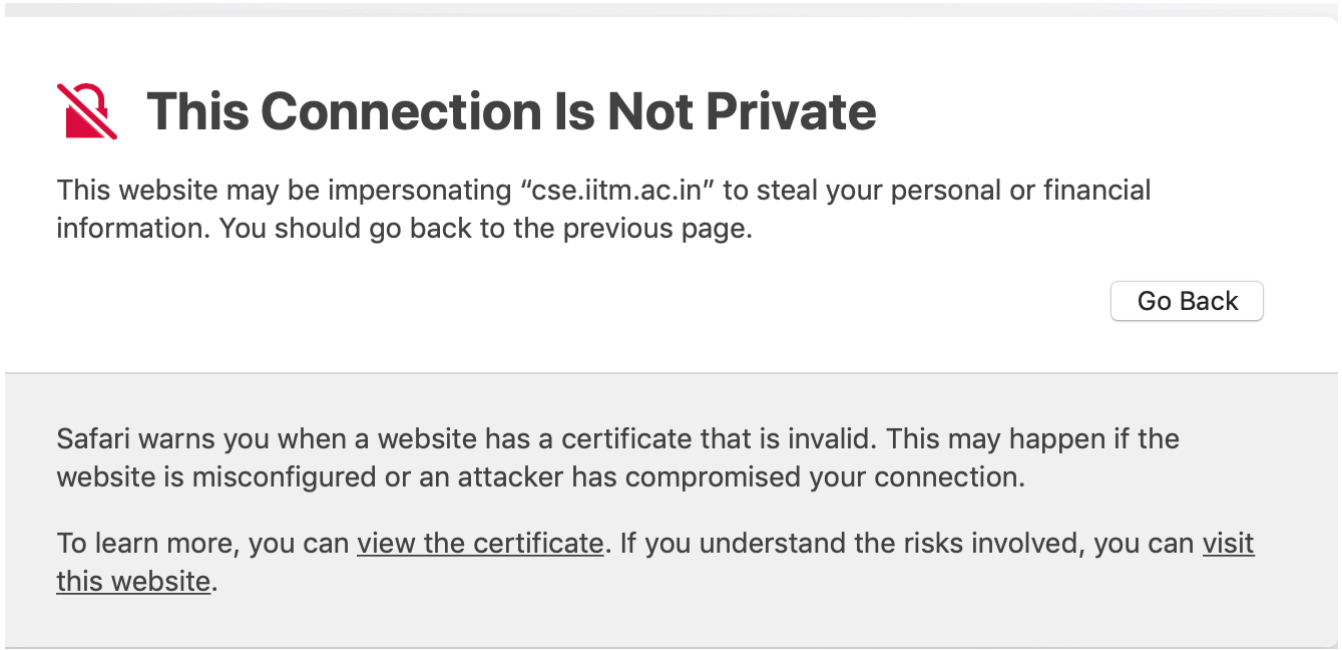
```
    Start Time: 1553014904
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
---
```

https://localhost:44330

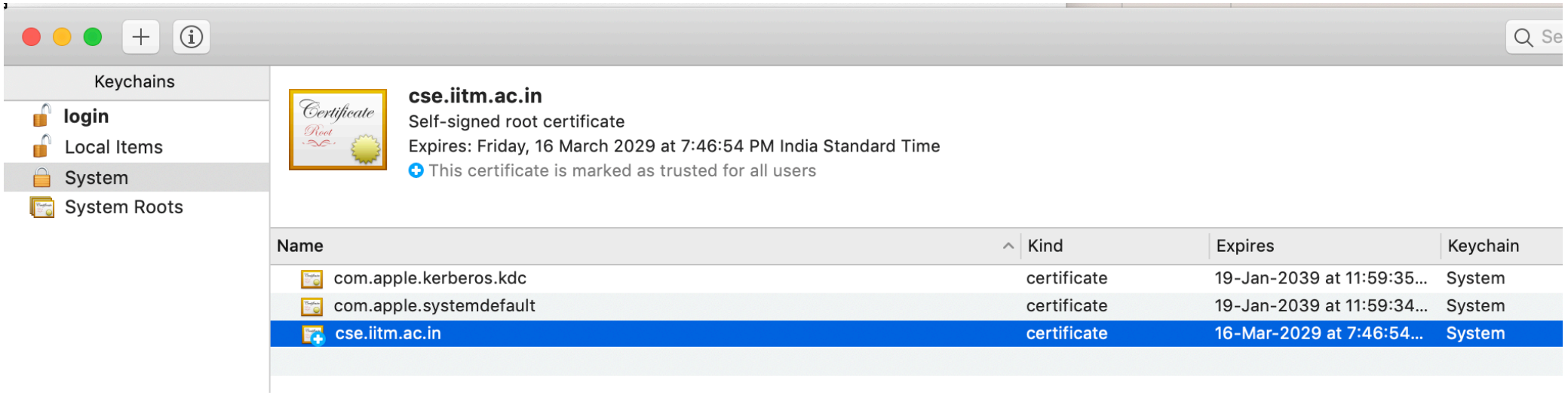




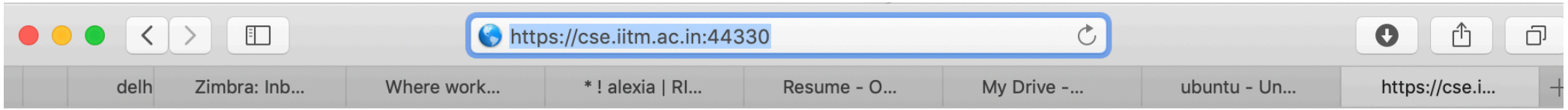
https://cse.iitm.ac.in:44330



Register modeCA in your system
(need to select that you trust this CA)

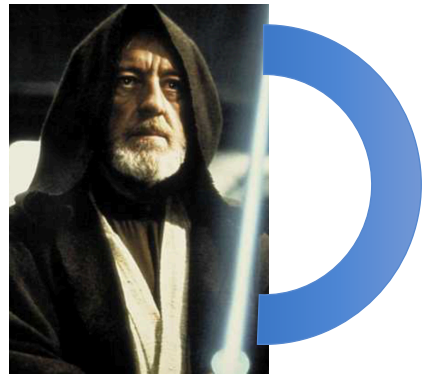


https://cse.iitm.ac.in:44330




```
s_server -key bank_key.pem -cert bank_cert.pem -accept 44330 -www
Secure Renegotiation IS supported
Ciphers supported in s_server binary
TLSv1/SSLv3:ECDHE-ECDSA-CHACHA20-POLY1305TLSv1/SSLv3:ECDHE-RSA-CHACHA20-POLY1305
TLSv1/SSLv3:DHE-RSA-CHACHA20-POLY1305TLSv1/SSLv3:ECDHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:ECDHE-ECDSA-AES256-GCM-SHA384TLSv1/SSLv3:ECDHE-RSA-AES256-SHA384
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA384TLSv1/SSLv3:ECDHE-RSA-AES256-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES256-SHA    TLSv1/SSLv3:DHE-RSA-AES256-GCM-SHA384
TLSv1/SSLv3:DHE-RSA-AES256-SHA256     TLSv1/SSLv3:DHE-RSA-AES256-SHA
TLSv1/SSLv3:GOST2012256-GOST89-GOST89TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA256
TLSv1/SSLv3:DHE-RSA-CAMELLIA256-SHA   TLSv1/SSLv3:GOST2001-GOST89-GOST89
TLSv1/SSLv3:AES256-GCM-SHA384         TLSv1/SSLv3:AES256-SHA256
TLSv1/SSLv3:AES256-SHA                TLSv1/SSLv3:CAMELLIA256-SHA256
TLSv1/SSLv3:CAMELLIA256-SHA           TLSv1/SSLv3:ECDHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-GCM-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA256
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA256TLSv1/SSLv3:ECDHE-RSA-AES128-SHA
TLSv1/SSLv3:ECDHE-ECDSA-AES128-SHA    TLSv1/SSLv3:DHE-RSA-AES128-GCM-SHA256
TLSv1/SSLv3:DHE-RSA-AES128-SHA256     TLSv1/SSLv3:DHE-RSA-AES128-SHA
TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA256TLSv1/SSLv3:DHE-RSA-CAMELLIA128-SHA
TLSv1/SSLv3:AES128-GCM-SHA256         TLSv1/SSLv3:AES128-SHA256
TLSv1/SSLv3:AES128-SHA                TLSv1/SSLv3:CAMELLIA128-SHA256
TLSv1/SSLv3:CAMELLIA128-SHA           TLSv1/SSLv3:ECDHE-RSA-RC4-SHA
TLSv1/SSLv3:ECDHE-ECDSA-RC4-SHA       TLSv1/SSLv3:RC4-SHA
TLSv1/SSLv3:RC4-MD5                   TLSv1/SSLv3:ECDHE-RSA-DES-CBC3-SHA
TLSv1/SSLv3:ECDHE-ECDSA-DES-CBC3-SHA TLSv1/SSLv3:EDH-RSA-DES-CBC3-SHA
TLSv1/SSLv3:DES-CBC3-SHA              TLSv1/SSLv3:EDH-RSA-DES-CBC-SHA
TLSv1/SSLv3:DES-CBC-SHA
```

Attacker forwards authentic certificate



2. Forward Request as is

5. Forward Request as is

3, Verify Identity of Alice
4. Digitally Sign

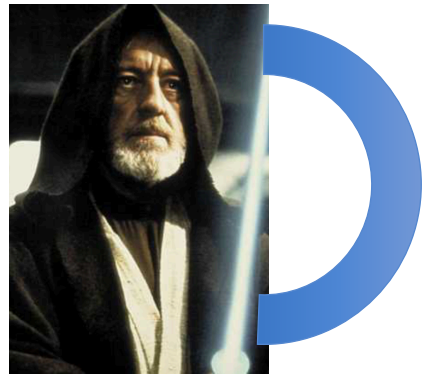
1. Register with CA

6. Forward Certificate as is

Bank.com

Attacker changes public key with her own

Request at CA is going to be fail
Because signature does not match public key



2. Forward Request (change public key)

5. Forward Request as is

3, Verify Identity of Alice
4. Digitally Sign

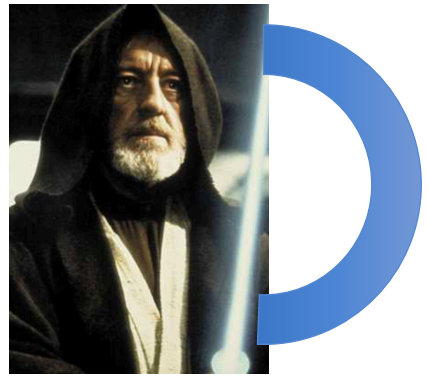
1. Register with CA

6. Forward Certificate as is

Bank.com

Attacker sends her own public key + signature

Verify should fail



2. Makes a request With bank's name

5. Forward Request as is

3, Verify Identity of Alice
4. Digitally Sign

1. Register with CA

6. Forward Certificate as is

Bank.com

Consider this Situation

Alice's certificate Signed by CA

Bank' s certificate

Bank.com Certificate Signed by CA

1. Attacker modifies public keys
2. Attacker replaces Bob's certificate with his/her own

Consider this Situation

Alice's certificate Signed by CA

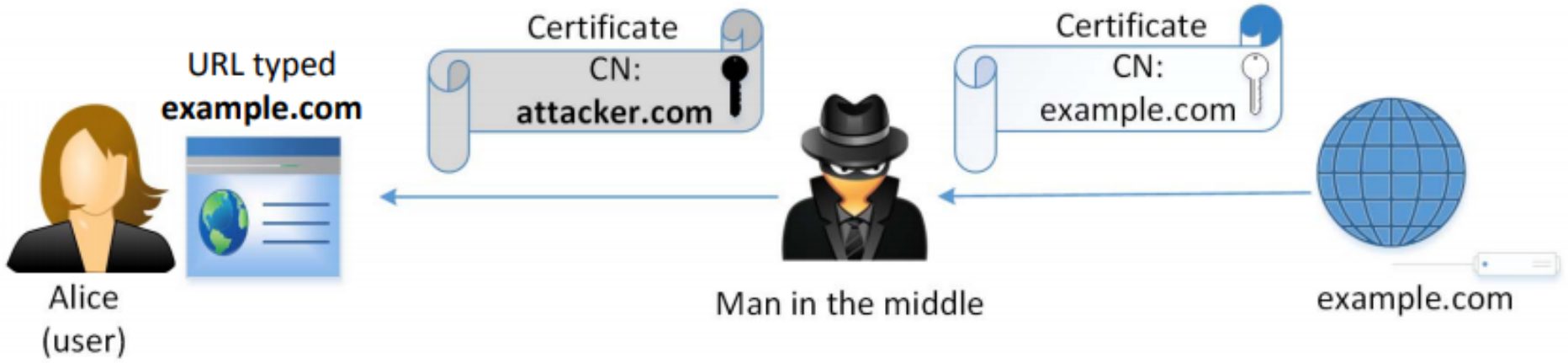
Bank's certificate

Bank.com Certificate Signed by CA

1. Attacker forwards fake certificate
2. Attacker replaces Bob's certificate with his/her own

(What is the requirement to have a MIMA?)

Attacker Sends His/Her Own Certificate



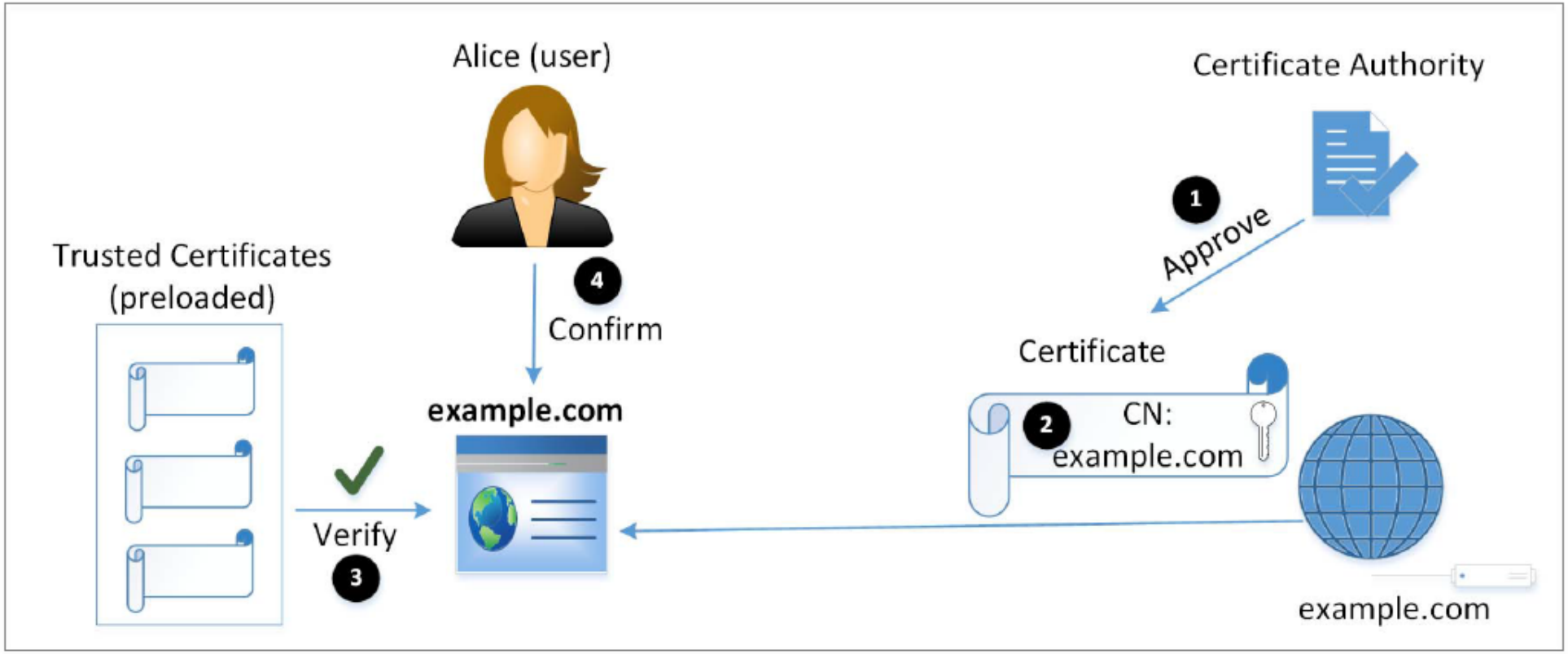
- Attacker's certificate is valid.
- Browser checks if the identity specified in the subject field of the certificate matches the Alice's intent.
  - There is a mismatch: attacker.com ≠ example.com
- Browser terminates handshake protocol: **MITM fails**

Emulating an MITM Attack

- DNS Attack is a typical approach to achieve MITM
  - We emulate an DNS attack by manually changing the /etc/hosts file on the user's machine to map example.com to the IP address of the attacker's machine.
- On attacker's machine we host a website for example.com.
  - We use the attacker's X.509 certificate to set up the server
  - The Common name field of the certificate contains attacker32.com
- When we visit example.com, we get an error message:

```
example.com uses an invalid security certificate.
The certificate is only valid for attacker32.com
(Error code: ssl_error_bad_cert_domain)
```

Attacks Surfaces on PKI



Attack on CA's Verification Process

- CA's job has two parts:
  - Verify the relationship between certificate applicant and the subject information inside the certificate
  - Put a digital signature on the certificate

- **Case study: Comodo Breach [March 2011]**
  - Popular root CA.
  - The approval process in Southern Europe was compromised.
  - Nine certificates were issued to seven domains and hence the attacker could provide false attestation.
  - One of the affected domain (a key domain for the Firefox browser): `addons.mozilla.org`

Attack on CA's Signing Process

- If the CA's private key is compromised, attackers can sign a certificate with any arbitrary data in the subject field.

- **Case Study: the DigiNotar Breach [June-July 2011]**
  - A top commercial CA
  - Attacker got DigiNotar's private key
  - 531 rogue certificates were issued.
  - Traffic intended for Google subdomains was intercepted: MITM attack.

- How CAs Protect Their Private Key
  - Hardware Security Model (HSM)

Attacks on Algorithms

- Digital Certificates depend on two types of algorithms
  - one-way hash function and digital signature

- **Case Study: the Collision-Resistant Property of One-Way Hash**
  - At CRYPTO2004, Xiaoyun Wang demonstrated collision attack against MD5.
  - In February 2017, Google Research announced SHAttered attack
    - Attack broke the collision-resistant property of SHA-1
    - Two different PDF files with the same SHA-1 has was created.

- Countermeasures: use stronger algorithm, e.g. SHA256.

Attacks on User Confirmation

- After verifying the certificate from the server, client software is sure that the certificate is valid and authentic
- In addition, the software needs to confirm that the server is what the user intends to interact with.
- Confirmation involves two pieces of information
  - Information provided or approved by user
  - The common name field inside the server's certificate
  - Some software does not compare these two pieces of information: security flaw

Attacks on Confirmation: Case Study

Phishing Attack on Common Name with Unicode

- Zheng found out several browsers do not display the domain name correctly if name contains Unicode.
- `xn–80ak6aa92e.com` is encoded using Cyrillic characters. But domain name displayed by browser likes like `apple.com`
- Attack:
  - Get a certificate for `xn–80ak6aa92e.com`
  - Get user to visit `xn–80ak6aa92e.com`, so the common name is matched
  - User's browser shows that the website is apple.com. **User can be fooled**.
- Had the browser told the user that the actual domain is not the real apple.com, the user would stop.