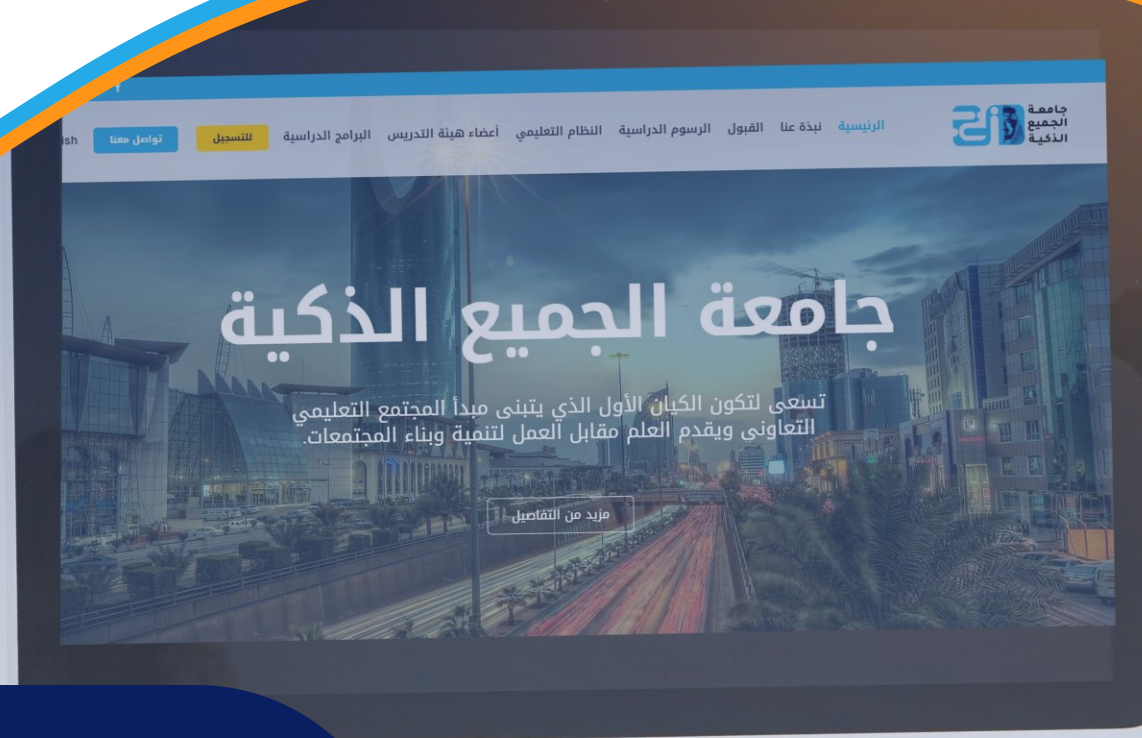




جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

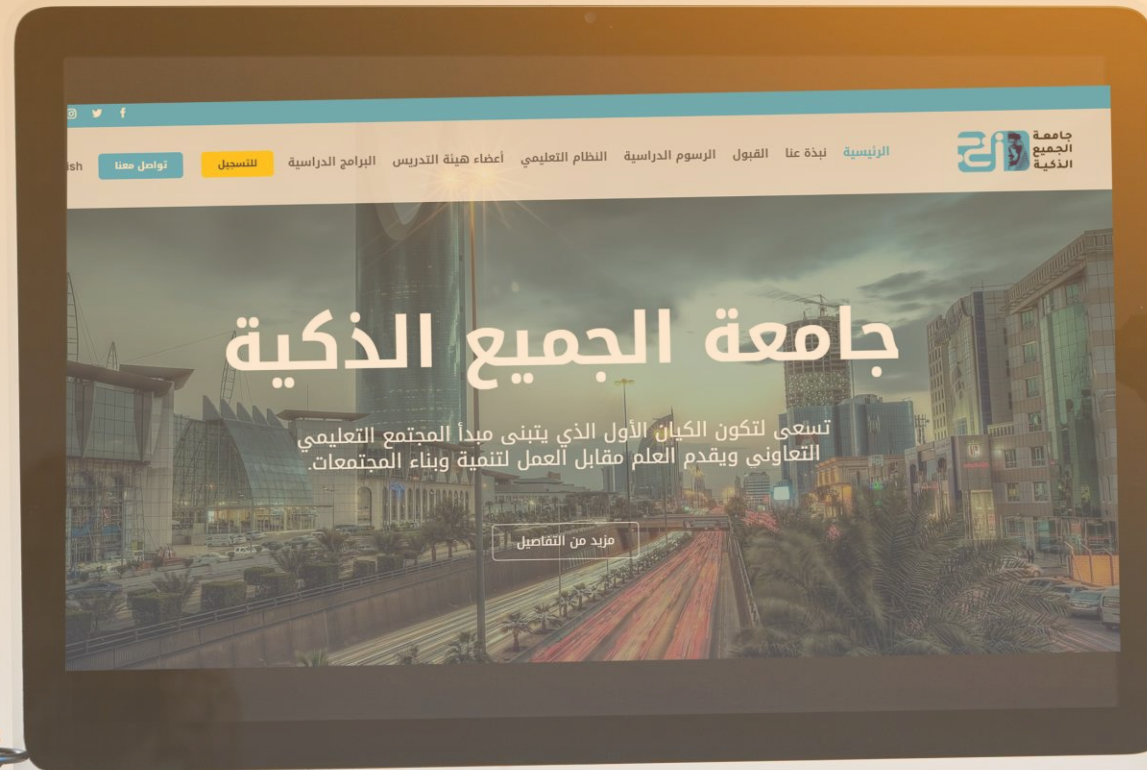


Prepared by
Dr. Hamada I. AbdulWakel
Lecture 6

CYB615
Network Programming



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY



Module4: Socket Programming using Python – Part III

- Broadcasting Client/Server Model.
- Securing Sockets.
- SSL Module.



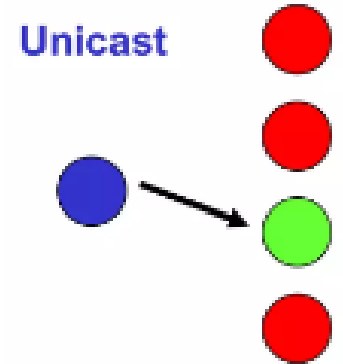
- Many applications transmit the same data at one time to multiple receivers:
 - Broadcasts of Radio or Video.
 - Video conferencing.
 - Shared applications.
- The **cast** term signifies some data (stream of packets) is being transmitted to the client(s) from a server over the communication channel.



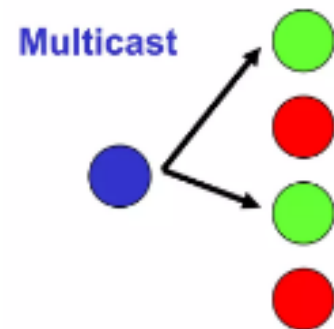
- There are three casting types:
 - Unicasting.
 - Multicasting.
 - Broadcasting.



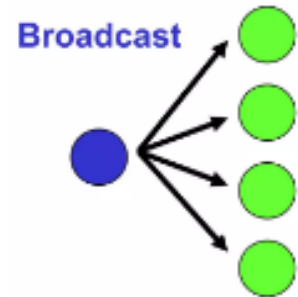
- Unicasting.
 - One-to-one communication.
 - One device on the network to another device on the network.
 - Example:
 - Browsing a website.
 - Downloading a file from a FTP server.



- Multicasting.
 - One-to-many communication.
 - One device on the network to many, but not all, devices on the network.
 - Devices which are interested in a particular multicast must join to that a **multicast group** to receive data.
 - Multicasting only supports UDP communication protocol.
 - Example:
 - Sending an email to a mailing list.



- Broadcasting.
 - One-to-all communication.
 - One device on the network to all devices on the network.
 - Network should be a local network.
 - Example:
 - Video & Radio.



Lecture 6

Broadcasting Client/Server UDP Example

```
import socket, threading, queue
messages = queue.Queue()
clients = []
server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
port = 9999
server.bind( ("localhost", port) )
def receive():
    while True:
        message, addr = server.recvfrom(1024)
        messages.put((message, addr))
```



Lecture 6

Broadcasting Client/Server UDP Example

Cont'd

```
def broadcast():
    while True:
        while not messages.empty():
            message, addr = messages.get()
            print(message.decode())
            if addr not in clients: clients.append(addr)
            for client in clients:
                try:
                    if message.decode().startswith("SIGNUP_TAG:"):
                        name = message.decode()[message.decode().index(":")+1 : ]
                        server.sendto(f"{name} joined!".encode(), client)
                    else:
                        server.sendto(message, client)
                except:
                    clients.remove(client)
t1 = threading.Thread(target=receive); t2 = threading.Thread(target=broadcast); t1.start(); t2.start()
```



```
import socket, threading, random
```

```
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  
client.bind( ("localhost", random.randint(8000, 9000)) )  
name = input("Your name: ")
```

```
def receive():  
    while True:  
        message, _ = client.recvfrom(1024)  
        print(message.decode())
```



```
t = threading.Thread(target=receive)
t.start()
```

```
client.sendto(f"SIGNUP_TAG:{name}".encode(), ("localhost", 9999))
```

```
while True:
```

```
    message = input()
```

```
    if message == "Q":
```

```
        break
```

```
    else:
```

```
        client.sendto(f"{name}: {message}".encode(), ("localhost", 9999))
```



- Network security is important for office, home, and business networks.
- **Network Attacker** completely owns the network:
 - Controls Wi-Fi, DNS, routers, his own websites, can listen to any packet, modify packets in transit, inject his own packets into the network.



- **Cipher:** an algorithm for performing encryption or decryption.
- **Encryption/Decryption:** the process of converting human-readable plaintext to incomprehensible text, also known as ciphertext.
- **Hashing:** the process of transforming any given key or a string of characters into another value. To know if the data is modified for not.
- **Symmetric/Asymmetric:**
 - In Symmetric-key encryption the message is encrypted by using a key and the same key is used to decrypt the message.
 - In Asym.-key encryption it based on public and private key encryption techniques.



- Widely deployed security protocol
 - Supported by almost all browsers and web servers.
- SSL provides:
 - Confidentiality.
 - Integrity.
 - Authentication.
- Goals:
 - Web E-commerce transactions.
 - Encryption (especially credit-card numbers)).
 - Web-server authentications.



- **Handshake:** A and B use their certificates, private keys to authenticate each other and exchange shared secret.
- **Key derivation:** A and B use shared secret to derive set of keys.
 - Considered bad to use same key for more than one cryptographic operation.
- **Data transfer:** data to be transferred is broken up into series of records.
- **Connection closure:** special messages to securely close connection.



- Install the OpenSSL toolkit.
- Create the Private Key and Certificate Signing Request (CSR) files.
- Submit CSR to your certificate authority (CA) of choice (e.g., GoDaddy, Verisign).
- Retrieve your certificate from the certificate authority.
- Download the “root certificate” from the same certificate authority.

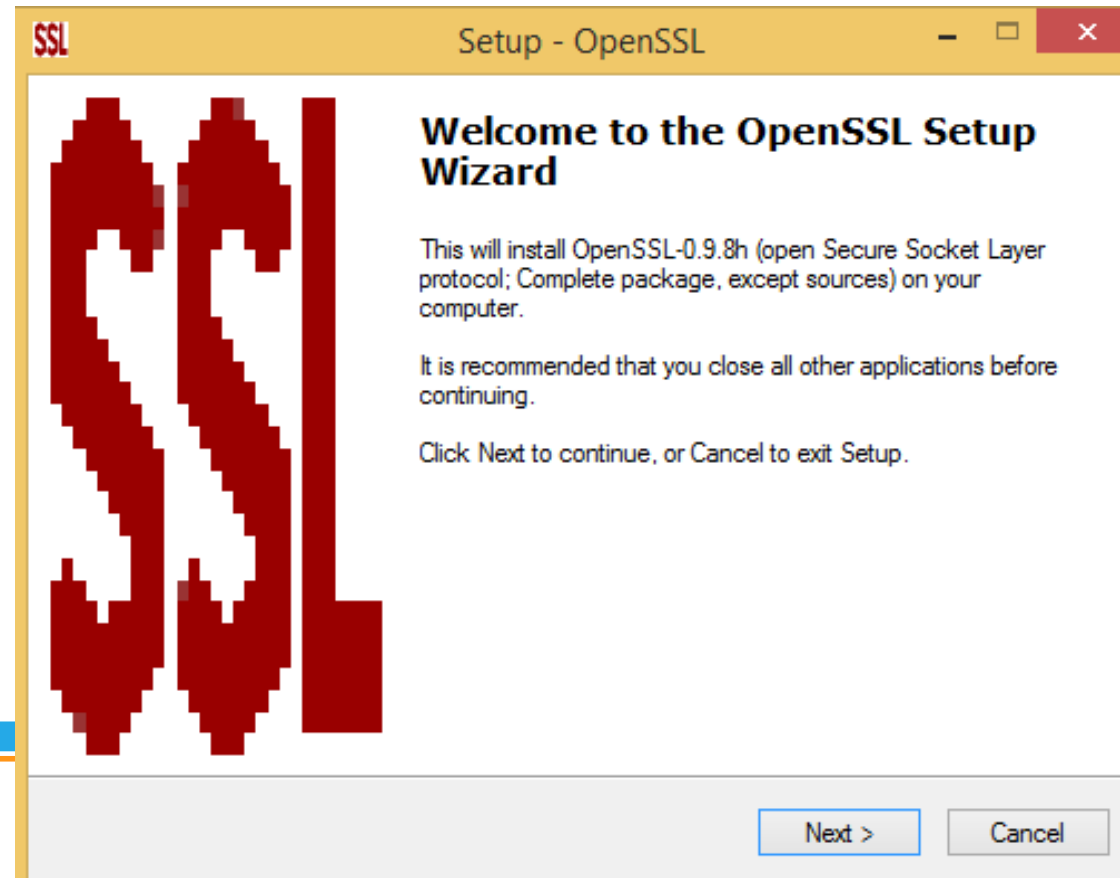


Lecture 6

OpenSSL Toolkit

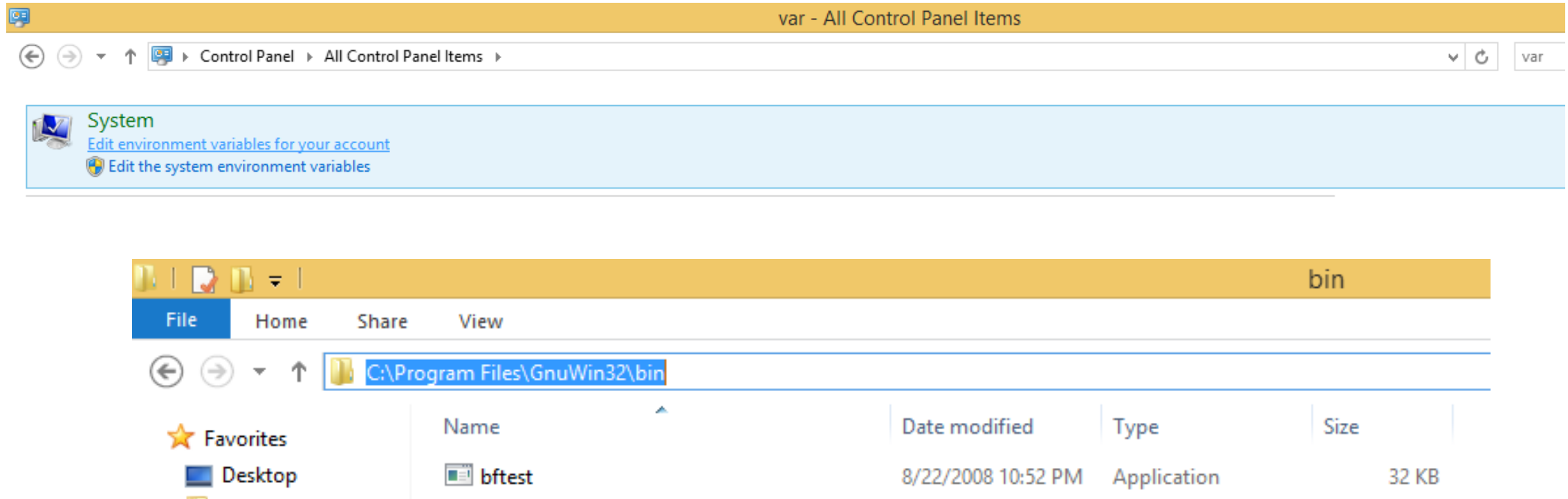
```
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.

C:\Users\Drhamada>openssl genrsa -des3 -out domain.key 2048
'openssl' is not recognized as an internal or external command,
operable program or batch file.
```



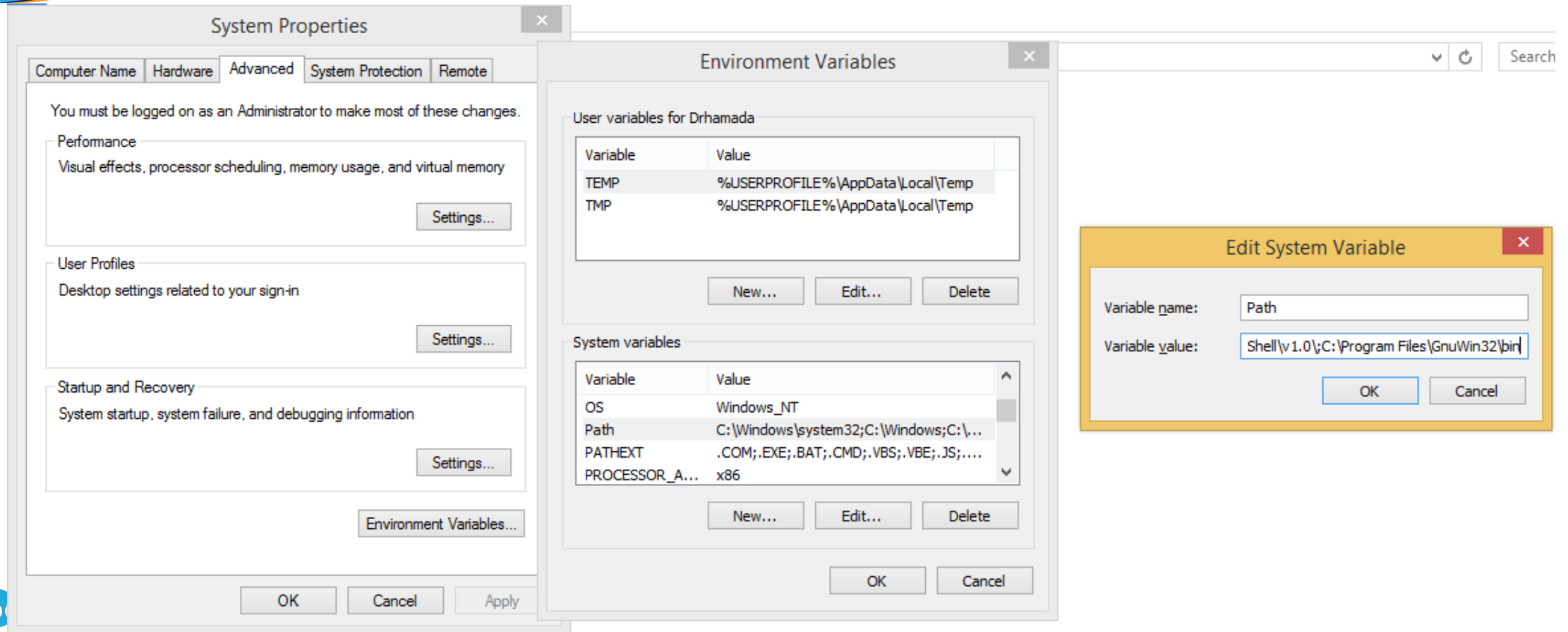
Lecture 6

OpenSSL Toolkit



Lecture 6

OpenSSL Toolkit



Lecture 6

OpenSSL Toolkit

```
Command Prompt
Microsoft Windows [Version 6.3.9600]
(c) 2013 Microsoft Corporation. All rights reserved.
C:\Users\Drhamada>openssl version
OpenSSL 0.9.8h 28 May 2008
```

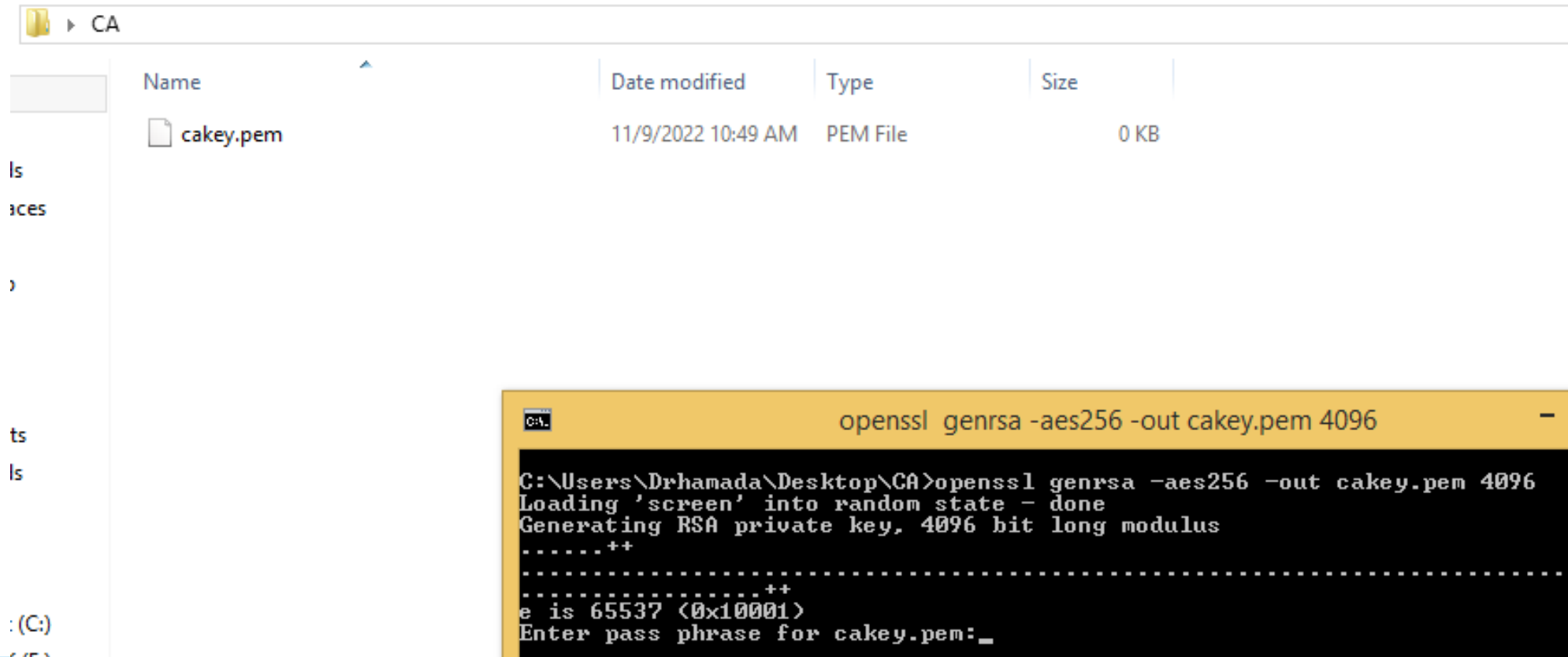


- Create a CA folder.
- openssl genrsa -aes256 -out caKey.pem 4096
- openssl req -new -x509 -sha256 -days 365 -key caKey.pem -out caCert.pem
- openssl genrsa -out certKey.pem 4096
- openssl req -new -sha256 -subj "/CN=<NAME>" -key certKey.pem -out certQuery.csr
- echo subjectAltName=IP:127.0.0.1 > extfile.cnf
- openssl x509 -req -sha256 -days 365 -in certQuery.csr -CA /path/to/caCert.pem -CAkey /path/to/caKey.pem -out certServer.pem -extfile extfile.cnf -CAcreateserial



Lecture 6

OpenSSL Toolkit



CA

Name	Date modified	Type	Size
cakey.pem	11/9/2022 10:49 AM	PEM File	0 KB

```
openssl genrsa -aes256 -out cakey.pem 4096
```

```
C:\Users\Drhamada\Desktop\CA>openssl genrsa -aes256 -out cakey.pem 4096
Loading 'screen' into random state - done
Generating RSA private key, 4096 bit long modulus
.....++
.....++
.....++
e is 65537 (0x10001)
Enter pass phrase for cakey.pem: _
```



Lecture 6

OpenSSL Toolkit

This PC > Local Disk (C:) > Program Files > GnuWin32 > share

Name	Date modified	Type	Size
certs	11/9/2022 10:39 AM	File folder	
misc	11/9/2022 10:39 AM	File folder	
private	11/9/2022 10:39 AM	File folder	
openssl.cnf	8/19/2005 1:38 PM	CNF File	10 KB

```
C:\Windows\system32\cmd.exe - openssl req -new -x509 -config "C:\Program ...  
  
C:\Users\Drhamada\Desktop\CA>openssl genrsa -aes256 -out cakey.pem 4096  
Loading 'screen' into random state - done  
Generating RSA private key, 4096 bit long modulus  
.....++  
.....++  
e is 65537 (0x10001)  
Enter pass phrase for cakey.pem:  
Verifying - Enter pass phrase for cakey.pem:  
  
C:\Users\Drhamada\Desktop\CA>openssl req -new -x509 -sha256 -days 365 -key cakey  
pem -out cacert.pem  
Unable to load config info from /usr/local/ssl/openssl.cnf  
  
C:\Users\Drhamada\Desktop\CA>openssl req -new -x509 -sha256 -days 365 -key cakey  
.pem -out cacert.pem  
Unable to load config info from /usr/local/ssl/openssl.cnf  
  
C:\Users\Drhamada\Desktop\CA>openssl req -new -x509 -config "C:\Program Files\Gn  
uWin32\share\openssl.cnf" -sha256 -days 365 -key cakey.pem -out cacert.pem  
Enter pass phrase for cakey.pem:
```



Lecture 6

OpenSSL Toolkit

```
C:\Windows\system32\cmd.exe - openssl req -new -x509 -config "C:\Program ...  
pem -out cacert.pem  
Unable to load config info from /usr/local/ssl/openssl.cnf  
  
C:\Users\Drhamada\Desktop\CA>openssl req -new -x509 -sha256 -days 365 -key cakey  
.pem -out cacert.pem  
Unable to load config info from /usr/local/ssl/openssl.cnf  
  
C:\Users\Drhamada\Desktop\CA>openssl req -new -x509 -config "C:\Program Files\Gn  
uWin32\share\openssl.cnf" -sha256 -days 365 -key cakey.pem -out cacert.pem  
Enter pass phrase for cakey.pem:  
Loading 'screen' into random state - done  
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]:eg  
State or Province Name (full name) [Some-State]:minia  
Locality Name (eg, city) []:minia  
Organization Name (eg, company) [Internet Widgits Pty Ltd]:  
Organizational Unit Name (eg, section) []:  
Common Name (eg, YOUR name) []:hamada  
Email Address []:abc@testcom_
```



Lecture 6

OpenSSL Toolkit

```
CA\ C:\Windows\system32\cmd.exe
file>
  -reqexts ..      specify request extension section (override value in config file)
)
  -utf8           input characters are UTF8 (default ASCII)
  -nameopt arg    - various certificate name options
  -reqopt arg     - various request text options

C:\Users\Drhamada\Desktop\CA>openssl req -new -sha256 -subj "\CN=hamada" -key ce
rtkey.pem -out certquery.csr
Unable to load config info from /usr/local/ssl/openssl.cnf

C:\Users\Drhamada\Desktop\CA>openssl req -new -config "C:\Program Files\GnuWin32
\share\openssl.cnf" -sha256 -subj "\CN=hamada" -key certkey.pem -out certquery.c
sr
Loading 'screen' into random state - done
Subject does not start with '/'.
problems making Certificate Request

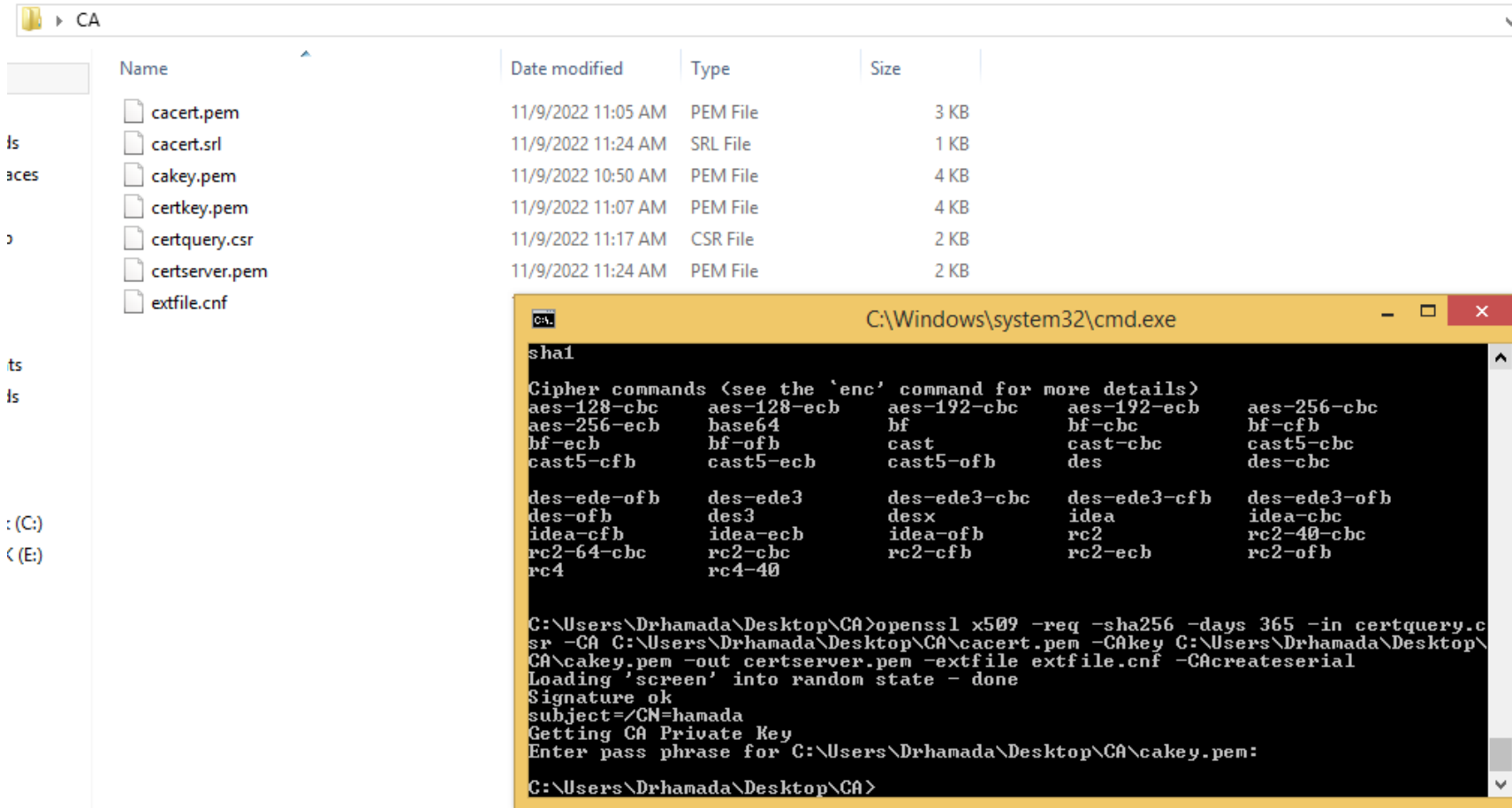
C:\Users\Drhamada\Desktop\CA>openssl req -new -config "C:\Program Files\GnuWin32
\share\openssl.cnf" -sha256 -subj "/CN=hamada" -key certkey.pem -out certquery.c
sr
Loading 'screen' into random state - done

C:\Users\Drhamada\Desktop\CA>_
```

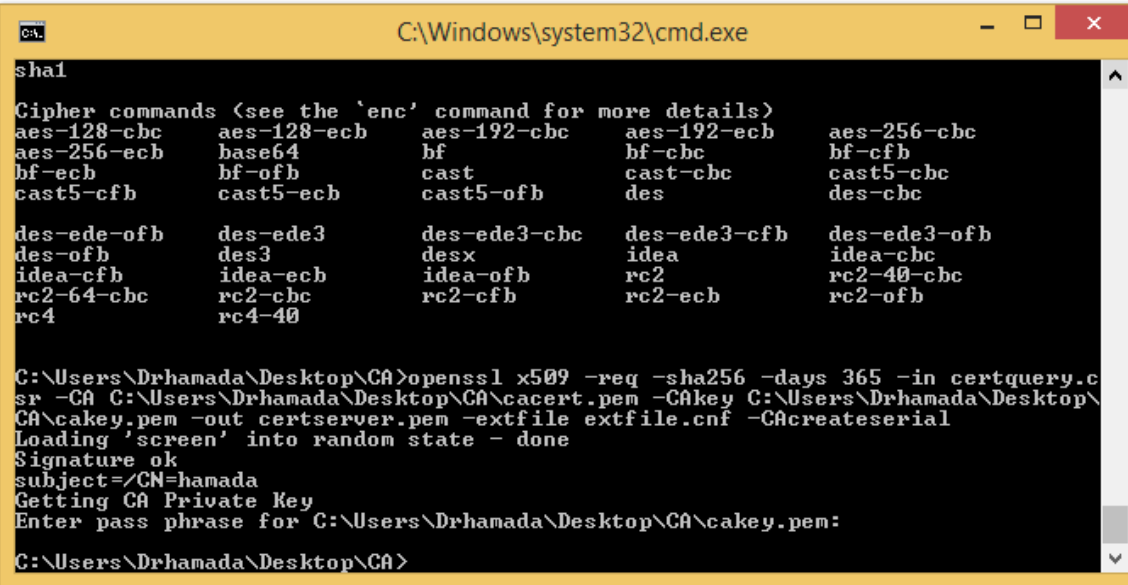


Lecture 6

OpenSSL Toolkit



Name	Date modified	Type	Size
cacert.pem	11/9/2022 11:05 AM	PEM File	3 KB
cacert.srl	11/9/2022 11:24 AM	SRL File	1 KB
cakey.pem	11/9/2022 10:50 AM	PEM File	4 KB
certkey.pem	11/9/2022 11:07 AM	PEM File	4 KB
certquery.csr	11/9/2022 11:17 AM	CSR File	2 KB
certserver.pem	11/9/2022 11:24 AM	PEM File	2 KB
extfile.cnf			



```
C:\Windows\system32\cmd.exe

sha1

Cipher commands (see the 'enc' command for more details)
aes-128-cbc aes-128-ecb aes-192-cbc aes-192-ecb aes-256-cbc
aes-256-ecb base64 bf bf-cbc bf-cfb
bf-ecb bf-ofb cast cast-cbc cast5-cbc
cast5-cfb cast5-ecb cast5-ofb des des-cbc

des-ede-ofb des-ede3 des-ede3-cbc des-ede3-cfb des-ede3-ofb
des-ofb des3 desx idea idea-cbc
idea-cfb idea-ecb idea-ofb rc2 rc2-40-cbc
rc2-64-cbc rc2-cbc rc2-cfb rc2-ecb rc2-ofb
rc4 rc4-40

C:\Users\Drhamada\Desktop\CA>openssl x509 -req -sha256 -days 365 -in certquery.csr -CA C:\Users\Drhamada\Desktop\CA\cacert.pem -CAkey C:\Users\Drhamada\Desktop\CA\cakey.pem -out certserver.pem -extfile extfile.cnf -CAcreateserial
Loading 'screen' into random state - done
Signature ok
subject=/CN=hamada
Getting CA Private Key
Enter pass phrase for C:\Users\Drhamada\Desktop\CA\cakey.pem:

C:\Users\Drhamada\Desktop\CA>
```



```
import socket, ssl
host = "192.168.1.7"; port = 10500
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#context creation
ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
ctx.load_cert_chain("/mnt/3126DF085226C5FC/Codes/CA/cert/cert-
server.pem", "/mnt/3126DF085226C5FC/Codes/CA/cert/cert-key.pem")
sock.bind((host, port)) #start server
sock.listen(5)
print("Server start")
```



#secure the server

```
sslServer = ctx.wrap_socket(sock, server_side=True)
```

#accept connection

```
client, ip = sslServer.accept()  
print("The IP", ip, " is connected")
```

#receive message

```
msg = client.recv(1024).decode()  
print("Client: ", msg)
```



Lecture 6

SSL Client/Server TCP Example

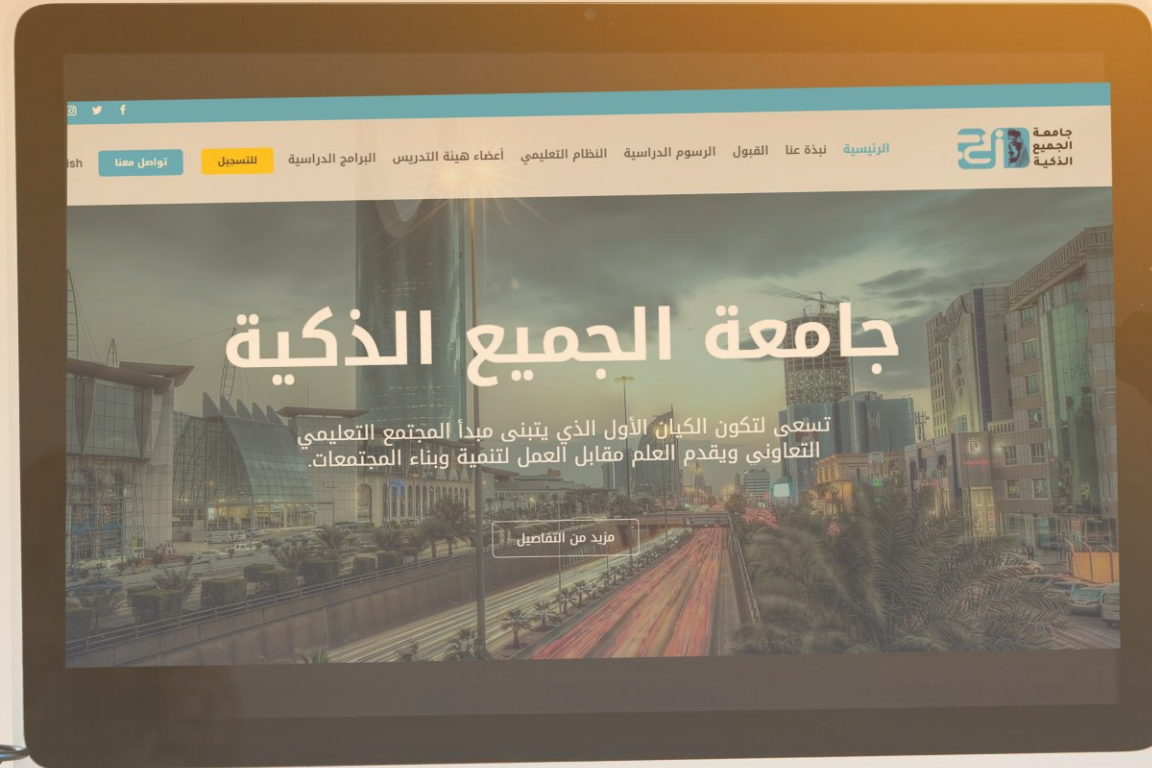
Cont'd

```
import socket, ssl
host = "192.168.1.7"; port = 10500
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT) #context creation
ctx.load_verify_locations("/mnt/3126DF085226C5FC/Codes/CA/ca-cert.pem")
sslClient = ctx.wrap_socket(sock, server_hostname=host) #secure the client
sslClient.connect((host, port))
msg = input(":> ").encode() #exchange message
sslClient.send(msg)
```



- Exercise 1:
Update the SSL TCP client/server code to send and receive messages until connection closed by the client.





جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

THANK YOU