

# Network Programming in Python

Prepared by Dr. Hamada I. AbdulWakel

## Lecture 6





مجموعة جامعة  
الجميع الذكية  
للتعليم القابضة

MIDOCEAN  
UNIVERSITY  
BY EVERYONE'S SMART UNIVERSITY

# Sockets Security in Python

## Lecture 6

Prepared by  
**Dr. Hamada I. AbdulWakel**

[www.midocean.ae](http://www.midocean.ae)

Lecture

06

## RECAP: LECTURE 5

- The threading programming.
- Programming the multiple client/server model.
- The casting/communication types.

Lecture

06

## LECTURE MAIN POINTS

- A broadcasting client/server example.
- Securing Sockets.
- The SSL Module.

## BROADCASTINGCLIENT/SERVER UDP EXAMPLE

- Code: 1- the server

```
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))
```

# BROADCASTING CLIENT/SERVER UDP EXAMPLE

Cont'd

- Code: 1- the server

```
def broadcast():  
    while True:  
        while not messages.empty():  
            message, addr = messages.get()  
            print(message.decode())  
            if addr not in clients:  
                clients.append(addr)
```

## BROADCASTING CLIENT/SERVER UDP EXAMPLE

Cont'd

- Code: 1- the server

```
def broadcast():  
    for client in clients: #inside second while after if statement  
        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)
```

## BROADCASTING CLIENT/SERVER UDP EXAMPLE

Cont'd

- Code: 1- the server

```
t1 = threading.Thread(target=receive)
t2 = threading.Thread(target=broadcast)
t1.start()
t2.start()
```

## BROADCASTING CLIENT/SERVER UDP EXAMPLE

Cont'd

- Code: 2- the client

```
import socket, threading, random
client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.bind( ("localhost", random.randint(8000, 9000)) )
name = input("Your name: ")
flag = False
def receive():
    while True:
        if flag == True: break
        message, _ = client.recvfrom(1024); print(message.decode())
```

## BROADCASTING CLIENT/SERVER UDP EXAMPLE

Cont'd

- Code: 2- the client

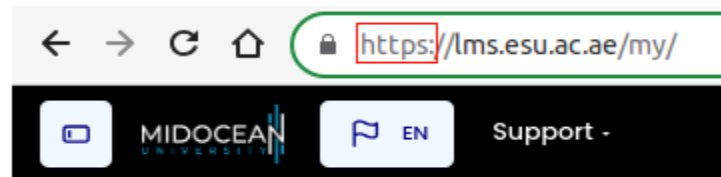
```
t = threading.Thread(target=receive) #no indentation
t.start()
client.sendto(f"SIGNUP_TAG:{name}".encode() , ("localhost", 9999))
while True:
    message = input()
    if message == "Q":
        client.close(); flag = True; break
    else:
        client.sendto(f"{name}: {message}".encode() , ("localhost", 9999))
```

Lecture

06

# SECURING CONNECTIONS

- A website URL starts with http:// or https://.
- The "s" in https:// signifies users are visiting a secure website, and the connection is encrypted and protected.
- HTTPS combines HTTP (Hyper Transfer Protocol Secure) and SSL (Secure Socket Layer).



Course overview

## SECURING CONNECTIONS

Cont'd

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

## SECURITY TERMINOLOGIES

- **Encryption:** the process of converting human-readable plain text to incomprehensible text, also known as ciphertext.
- **Decryption:** the process of converting cipher text to a human-readable plain text.

# SECURITY TERMINOLOGIES

Cont'd

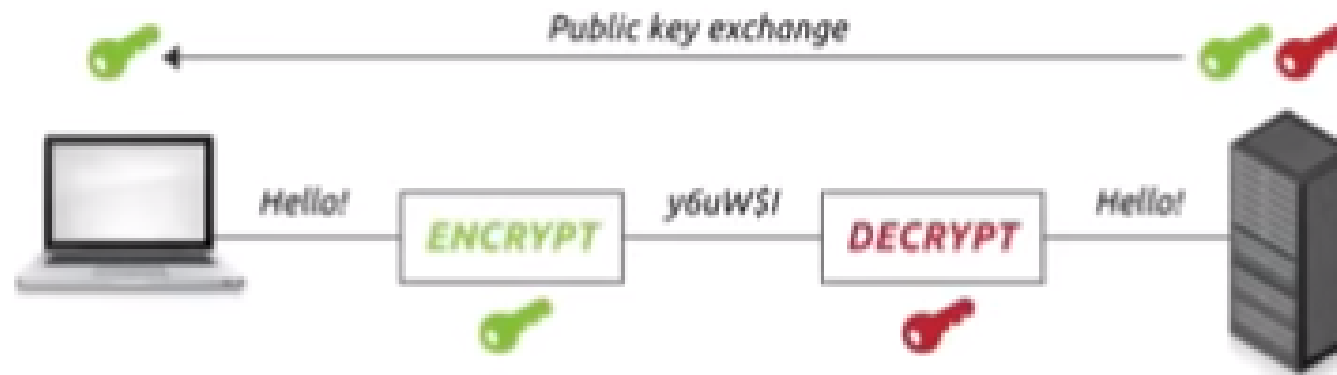
- Types of encryption:
  - **Symmetric-key encryption:** the message is encrypted and decrypted by using the same key.



## SECURITY TERMINOLOGIES

Cont'd

- Types of encryption:
  - Asymmetric-key encryption:** it based on public and private keys.



Lecture

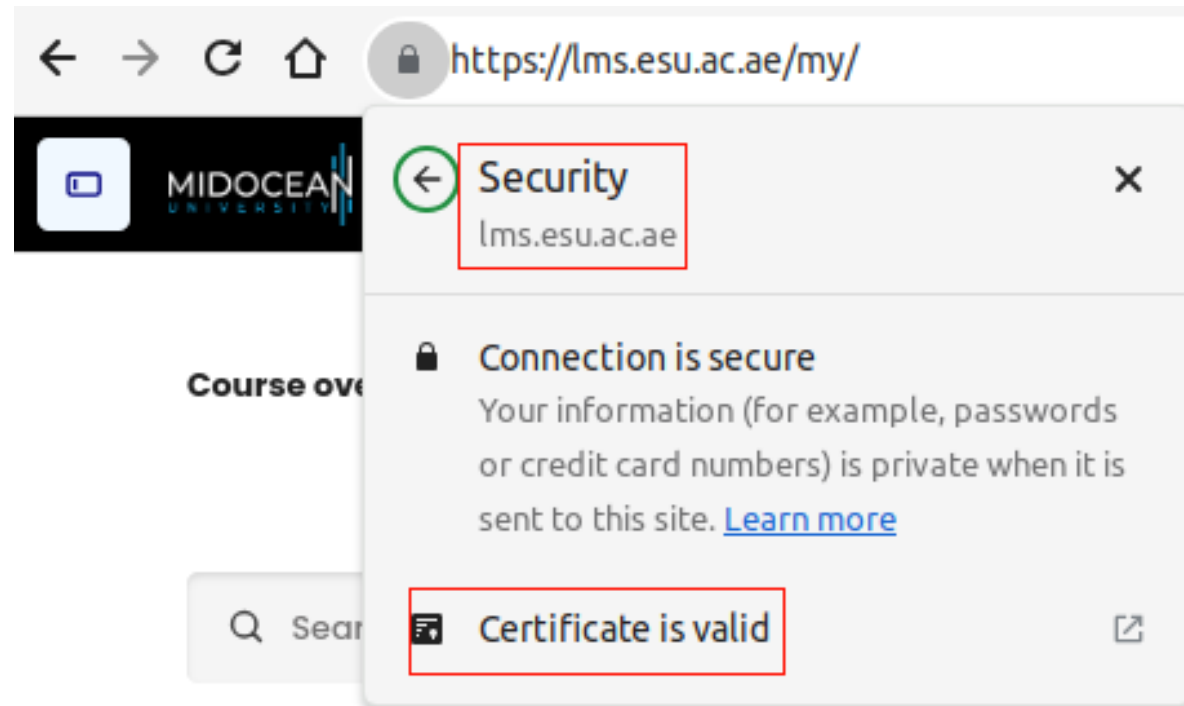
06

## SSL: SECURE SOCKEY LAYER

- A protocol used by SSL certificates to initiate secure sessions with a browser.
- Widely deployed security protocol that supported by almost all browsers and web servers.
- SSL certificates are small data files that use a public key infrastructure to make internet transactions secure when using HTTPs protocol.
- Goals:
  - Bank account details and credit card transactions.
  - Medical records.
  - E-commerce transactions encryption.

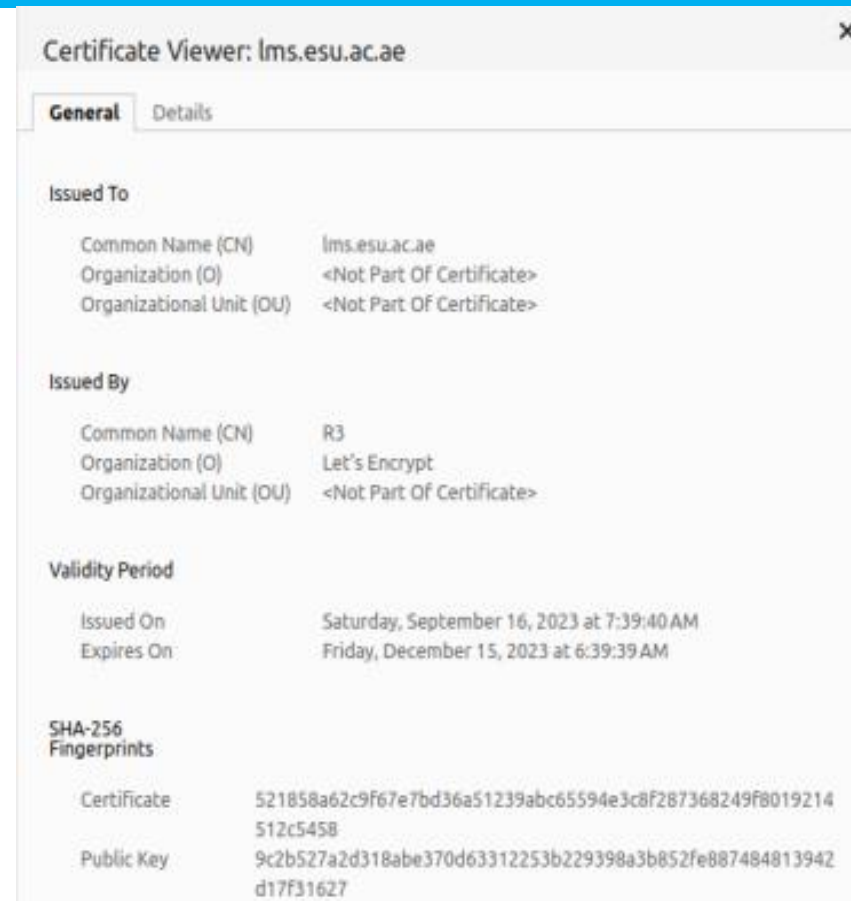
# SSL: SECURE SOCKEY LAYER

Cont'd

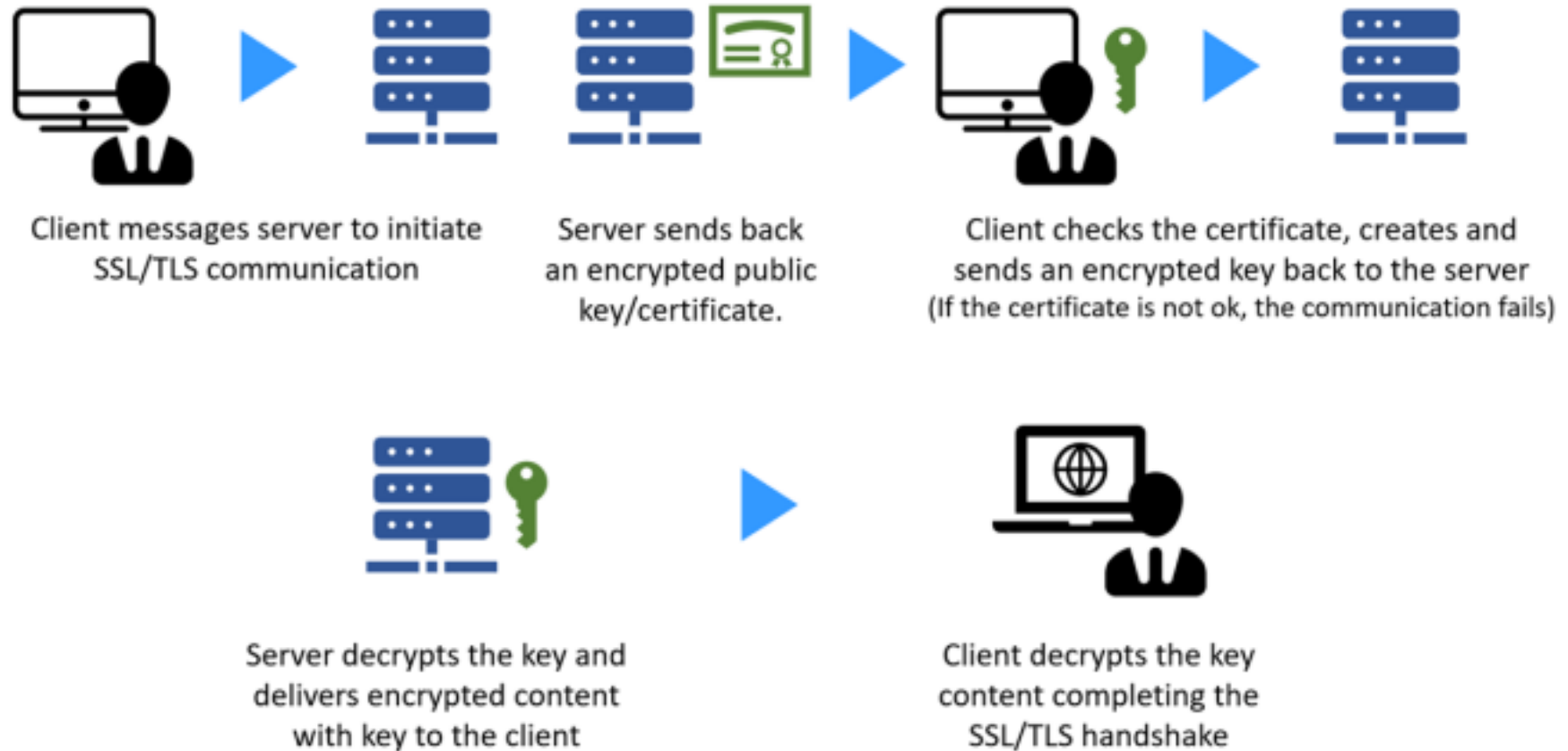


# SSL: SECURE SOCKEY LAYER

Cont'd



## SSL CLIENT/SERVER STEPS



## SSL CLIENT/SERVER STEPS

Cont'd

- When a client connects to a server secured with SSL, the secured server will send back an encrypted public key and a certificate.
- The client will then check the certificate and if it is trusted, the client will create an encrypted key using the server's public key and send the new key to the server.
- Lastly, the server will decrypt the new key using the server's private key and then send an acknowledgment to the client that is encrypted with the new key created by the client.
- Now, the server and the client can encrypt all transmissions with the new key.

## SSL CLIENT/SERVER STEPS

Cont'd

- **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.

## GETTING AN SSL CERTIFICATE

- To get an SSL certificate, you need to go through the following steps:
  - Start by verifying your website information and ensuring it is the same as the information you are submitting to the CA.
    - Certificate Authority (CA): a trusted entity that issues digital certificates which certifies the owner of a public key.
  - Then, generate the certificate signing request (CSR) through an online generator or with the help of a hosting company.
  - Submit the CSR to the certificate authority to validate the domain.
  - Once the process is complete, install and use the SSL certificate on your website.

Lecture

06

## OPENSSL TOOLKIT

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

# OPENSSL TOOLKIT

Cont'd



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

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

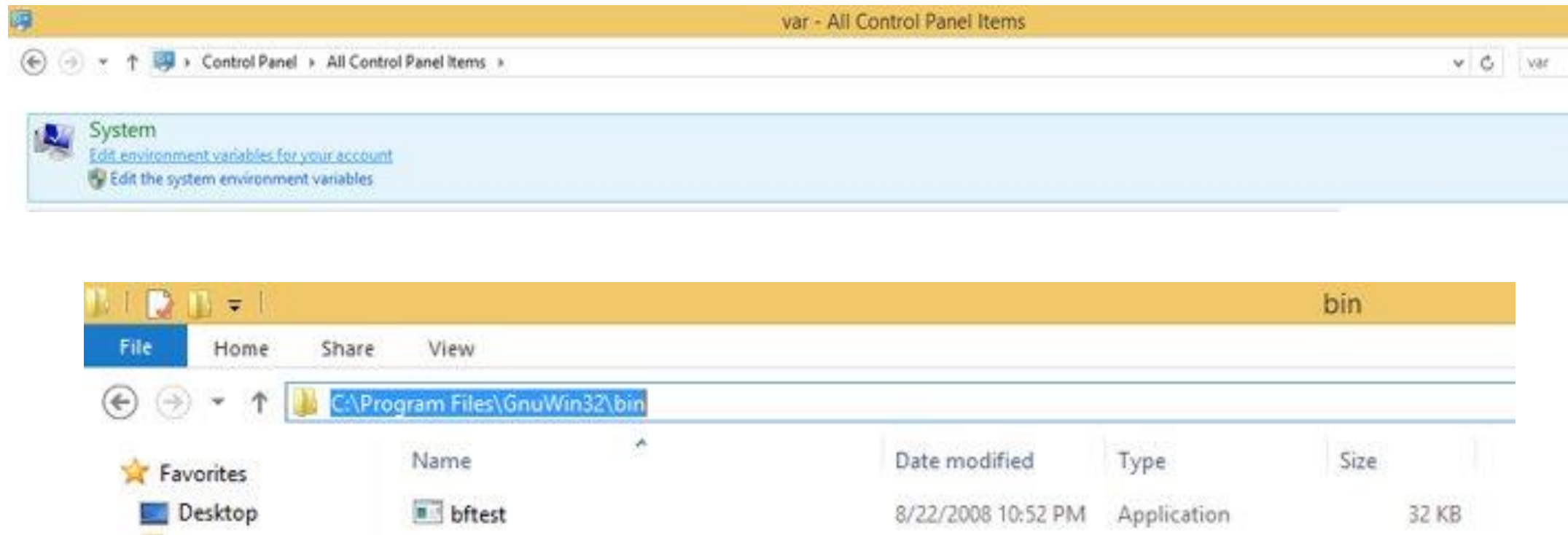


Lecture

06

# OPENSSL TOOLKIT

Cont'd

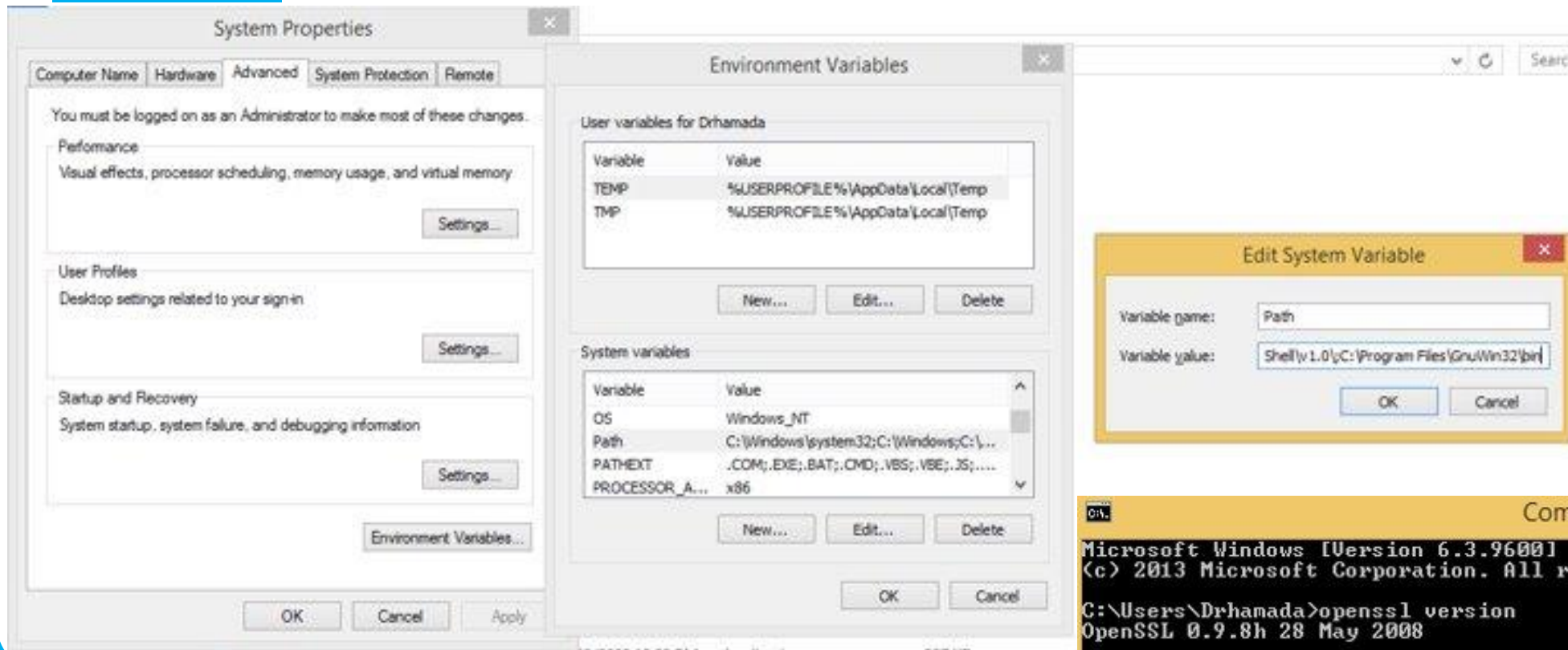


Lecture

06

# OPENSSL TOOLKIT

Cont'd



## SSL SELF-SIGNED CERTIFICATION

- 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
```

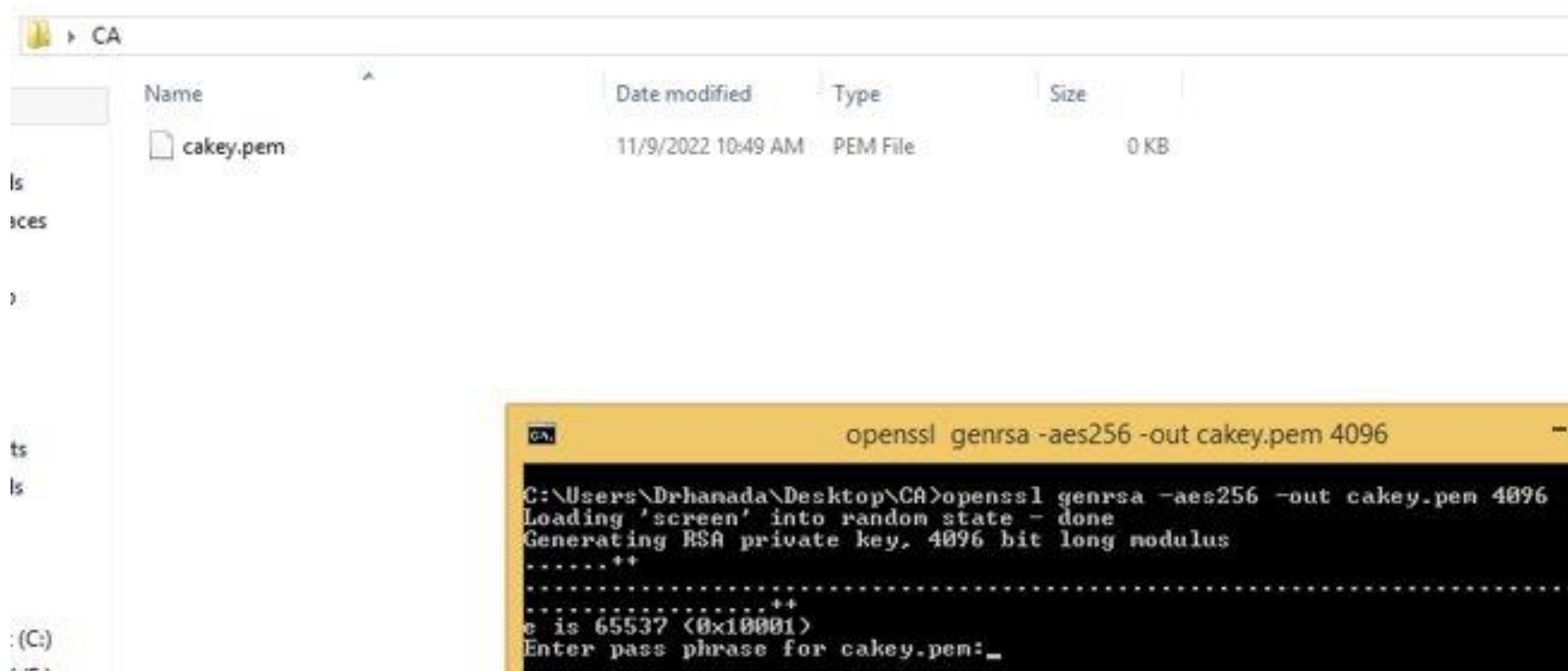
```
echo subjectAltName=IP:127.0.0.1 > extfile.cnf
```

```
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

06

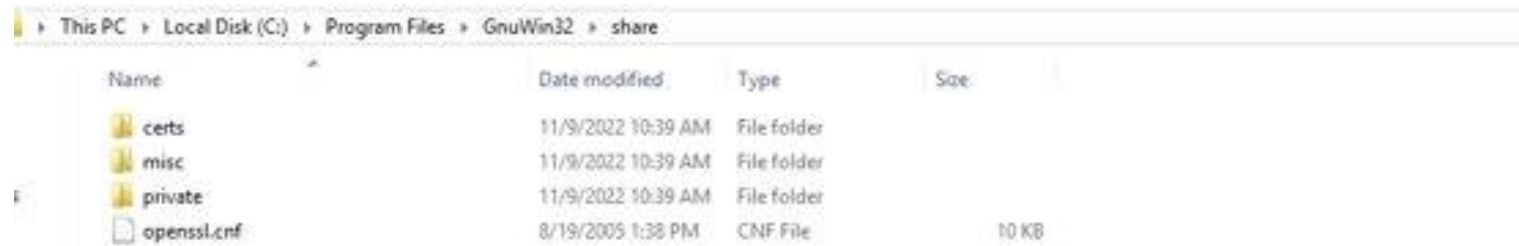
# SSL SELF-SIGNED CERTIFICATION



Lecture

06

# SSL SELF-SIGNED CERTIFICATION



```
C:\Windows\system32\cmd.exe - openssl req -new -x509 -config "C:\Program Files\GnuWin32\share\openssl.cnf" -sha256 -days 365 -key cakey.pem -out cacert.pem
C:\Users\Drhanada\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\Drhanada\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\Drhanada\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\Drhanada\Desktop\CA>openssl req -new -x509 -config "C:\Program Files\GnuWin32\share\openssl.cnf" -sha256 -days 365 -key cakey.pem -out cacert.pem
Enter pass phrase for cakey.pem:
```

Lecture

06

# SSL SELF-SIGNED CERTIFICATION

```
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

06

## TCP-SSL CLIENT/SERVER EXAMPLE

- Code: 1- the server

```
import socket, ssl
host = "127.0.0.1"
port = 10500
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#context creation
ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_SERVER)
ctx.load_cert_chain("/path/to/certServer.pem", "/path/to/certKey.pem")
sock.bind((host, port))
```

## TCP-SSL CLIENT/SERVER EXAMPLE

Cont'd

- Code: 1- the server

```
#start server
sock.listen()
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")
print("Client: ", client.recv(1024).decode() )
```

## TCP-SSL CLIENT/SERVER EXAMPLE

Cont'd

- Code: 2- the client

```
import socket, ssl
host = "127.0.0.1"
port = 10500
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#context creation
ctx = ssl.SSLContext(ssl.PROTOCOL_TLS_CLIENT)
ctx.load_verify_locations("/path/to/caCert.pem")
```

## TCP-SSL CLIENT/SERVER EXAMPLE

Cont'd

- Code: 2- the client

```
#secure the client
```

```
sslClient = ctx.wrap_socket(sock, server_hostname=host)
```

```
sslClient.connect((host, port))
```

```
#exchange message
```

```
msg = input(":> ").encode()
```

```
sslClient.send(msg)
```

Lecture

06

## QUESTIONS FOR ONLINE DISCUSSION

- Try to modify the SSL client/server program to allow the server to accept multiple clients and secure connections with them.



مجموعة جامعة  
الجميع الذكية  
للتعليم القابضة

MIDOCEAN  
UNIVERSITY  
BY EVERYONE'S SMART UNIVERSITY



# Thank You

n-gl.com

