

Laboratorio 5 evaluado de seguridad informática

Consideraciones importantes

- El laboratorio se puede hacer con un máximo de 2 personas
- Cualquier sospecha de copia será evaluada con la nota mínima 1.0 para todos los involucrados.
- Tiene una semana para entregarlo

Usted tiene permitido:

- Trabajar en conjunto a un compañero
- Utilizar el lenguaje que desee

Desafío: Se le solicita a usted crear un programa del tipo cliente / servidor, el cual tenga como objetivo que el cliente pueda sincronizar con el servidor una llave mediante diffie-helman para posteriormente recibir un el mensaje leído desde el archivo mensajeentrada.txt el cual debe ser encriptado por DES y poder desencriptar en nuevo archivo llamado mensajerecibido.txt

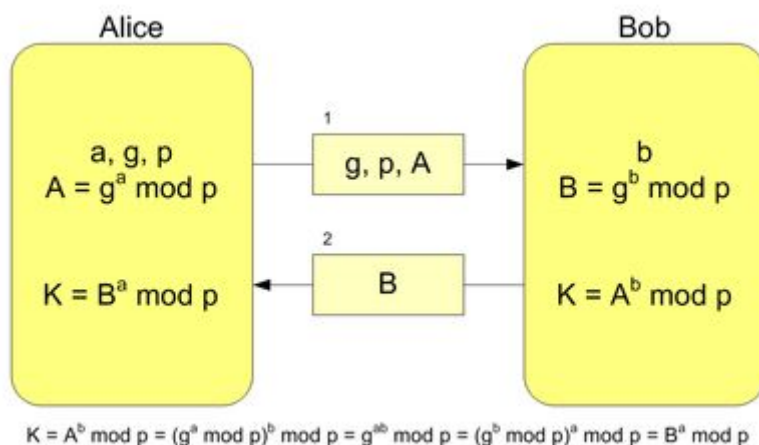
Al terminar la actividad repita lo mismo ocupando 3DES y aes con la cantidad de bytes que usted desee.

Entregable:

- Publicar su código en github
- Crear un video de no más de 5 minutos explicando su solución.

Desarrollo:

Se elaboró un programa cliente servidor para la sincronización de una llave.



A través de dos clases llamadas "alice" y "Bob":

```
class alice:
    def __init__(self,g,p,a):
        self.g=g
        self.p=p
        self.a=a

    def A(self):
        A=(self.g**self.a)%self.p
        return A

    def k(self,B):
        k=(B**self.a)%self.p
        return k

a=70
g=50
p=73

#B=55
uno=alice(g,p,a)
A=uno.A()
#k=uno.k(B)
print('A='+str(A))
#print('k='+str(k))

bhu=str(g)+','+str(p)+','+str(A)+','+(g;p;A)
print(bhu)

dos=receptor()
dos.respuesta_A(bhu)

dat=dos.contacto_B()[2:-1]
B=int(dat)
print('_____')
print("B'="+str(B))

k=uno.k(B)
print('k='+str(k))
```

```
class bob:
    def __init__(self,b):
        self.b=b
        self.g=None
        self.p=None
        self.A=None

    def datos(self,s):
        self.g=s[0]
        self.p=s[1]
        self.A=s[2]

    def B(self):
        B=(self.g**self.b)%self.p
        return(B)

    def k(self):
        k=(self.A**self.b)%self.p
        return k

b=20
s=[int(dat[0]),int(dat[1]),int(dat[2])]

dos=bob(b)
dos.datos(s)
B=dos.B()
k=dos.k()
print('B='+str(B))
print('k='+str(k))
```

Estas clases sincronizan una llave a través de diffie-hellman.

(<https://rico-schmidt.name/pydotw-3/socket/tcp.html>)

Además de otras dos clases que realizan la comunicación entre ambos interlocutores “emisor” y “receptor” para “bob” y “alice” respectivamente.

“emisor” con las funciones contacto_A y respuesta_B que realizan la comunicación

```
1 import socket
2 import sys
3 import pyDes
4
5 class emisor:
6     def contacto_A(self):
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43 def respuesta_B(self, tex):
44     # Create a TCP/IP socket
45     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
46
47     # Connect the socket to the port where the server is listening
48     server_address = ('localhost', 1000)
49     print('>>> connecting to {} port {}'.format(*server_address))
50     sock.connect(server_address)
51
52     try:
53         # Send data
54         a = tex
55         message = bytes(a, "ascii")
56         print('>>> sending {!r}'.format(message))
57         sock.sendall(message)
58
59         # Look for the response
60         amount_received = 0
61         amount_expected = len(message)
62
63         while amount_received < amount_expected:
64             data = sock.recv(16)
65             amount_received += len(data)
66             print('>>> received {!r}'.format(data))
67
68     finally:
69         print('>>> closing socket')
70         sock.close()
71
72 class bob:
```

“receptor” con las funciones contacto_B y respuesta_A que realizan la comunicación.

```

1 import socket
2 import sys
3 import pyDes
4
5 class receptor:
6     def contacto_B(self):
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43 def respuesta_A(self, tex):
44     # Create a TCP/IP socket
45     sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
46
47     # Connect the socket to the port where the server is listening
48     server_address = ('localhost', 10000)
49     print('>>> connecting to {} port {}'.format(*server_address))
50     sock.connect(server_address)
51
52     try:
53         # Send data
54         a=tex
55         message=bytes(a, "ascii")
56         #message = b'sd'
57         print('>>> sending {!r}'.format(message))
58         sock.sendall(message)
59
60         # Look for the response
61         amount_received = 0
62         amount_expected = len(message)
63
64         while amount_received < amount_expected:
65             data = sock.recv(16)
66             amount_received += len(data)
67             print('>>> received {!r}'.format(data))
68
69     finally:
70         print('>>> closing socket')
71         sock.close()
72
73 class alice:

```

Estas realizan la sincronización de la llave entre los dos interlocutores, “receptor empieza iniciado la comunicación con bob y después “emisor” define ‘a’, ‘g’ y ‘p’, para generar el ‘A’ que envía posteriormente a “receptor”, luego “emisor inicia una comunicación con “Alice” para que después que reciba ‘a’, ‘g’ y ‘p’, genere un ‘B’ que envía a de vuelta a “emisor”.