NSS Assignment

SAHIL BONDRE U18CO021

Implement the Needham-Schroeder Protocol including three entities Alice (A), Bob(B) and Key Distribution Center (KDC). Assume that there is secure communication between A and KDC, and B and KDC.

a) Register A to KDC where symmetric key is established between A and KDC.
b) Register B to KDC where symmetric key is established between B and KDC.
c) Generate session key between A and B via KDC.
d) A and B should send an acknowledgment message to each other once the session key is established.

Steps for Needham-Schroeder Protocol:

Step 1: $A \rightarrow KDC : A, B, N_A$

Step 2: $KDC \rightarrow A : \{N_A, K_{AB}, B, \{K_{AB}, A\}_{K_B}\}_{K_A}$

Step 3: $A \rightarrow B : \{K_{AB}, A\}_{K_B}$

Step 4: $B \rightarrow A : \{N_B\}_{K_{AB}}$

Step 5: $A \rightarrow B : \{N_B - 1\}_{K_{AB}}$

where K_A: Symmetric key known to A and KDC
K_B: Symmetric key known to B and KDC
K_{AB}: Session key between A and B
N_A: Nonce generated by A
N_B: Nonce generated by B

encrypt.py

```
from cryptography.fernet import Fernet

import json



class Cipher:
```

```
    def __init__(self, key):
        self.f = Fernet(key.encode('utf-8'))

    def encrypt(self, obj):
        return self.f.encrypt(json.dumps(obj).encode('utf-8')).decode()

    def decrypt(self, txt):
        return json.loads(self.f.decrypt(txt.encode('utf-8')))


def generate_key():
    return Fernet.generate_key().decode()
```

kdc.py

```
from encrypt import Cipher, generate_key
from threading import Thread
from queue import Queue
from time import sleep
from random import randint

def random_wait():
    # random wait between 1 and 3 seconds
    sleep(randint(50, 200) / 100)

ka = generate_key()
kb = generate_key()
```

```python
a_id = 100
b_id = 200

def kdc(kdc_alice_queue, ka, kb):
    a_cipher = Cipher(ka)
    b_cipher = Cipher(kb)
    print('KDC: Starting KDC server...')
    print('KDC: Waiting for Alice...')
    random_wait()

    # Step 1
    while kdc_alice_queue.empty():
        sleep(1)
    msg = kdc_alice_queue.get()
    print(f'KDC: Received from Alice Server {msg}')
    Na = msg[-1]

    # Step 2
    print()
    print('========== Step 2 KDC -> A (Na, Kab, B, (Kab, A)Kb)Ka ==========')
    input(f'KDC: Executing Step 2: Press enter to continue... ')
    kab = generate_key()
    payload = [Na, kab, b_id, b_cipher.encrypt([kab, a_id])]
    print(f'KDC: Payload {payload}')
    print(f'KDC: Kab: {kab}')
```

```
    txt = a_cipher.encrypt(payload)
    print(f'KDC: Encrypted Payload {txt}')
    input('KDC: Press Enter to send payload to Alice... ')
    kdc_alice_queue.put(txt)



def alice(kdc_alice_queue, alice_bob_queue, ka):
    a_cipher = Cipher(ka)
    print(f'ALICE: Starting Alice server (id: {a_id})...')
    random_wait()

    # Step 1
    print()
    print('========== Step 1 (A -> KDC (A, B, Na)) ==========')
    input('ALICE: Executing Step 1: Press enter to continue... ')
    Na = randint(1000, 9999)
    print(f'ALICE: Nonce: {Na}')
    kdc_alice_queue.put([a_id, b_id, Na])
    print('ALICE: Done')
    sleep(3)

    # Step 2
    while kdc_alice_queue.empty():
        sleep(1)

    msg = kdc_alice_queue.get()
```

```python
print()
print(f'ALICE: Received encrypted message from KDC {msg}')
input('ALICE: Press enter to decode... ')
msg = a_cipher.decrypt(msg)
print(f'ALICE: Decrypted: {msg}')
print(f'ALICE: Kab: {msg[1]}')
kab = msg[1]
ab_cipher = Cipher(kab)


# Step 3
print()
print('========== Step 3 (A -> B (Kab, A)Kb) ==========')
input('ALICE: Press enter to send encrypted keys to Bob... ')
alice_bob_queue.put(msg[3])
sleep(3)


# Step 4
while alice_bob_queue.empty():
    sleep(1)


msg = alice_bob_queue.get()
print()
print(f'ALICE: Received encrypted message from Bob {msg}')
input('ALICE: Press enter to decode... ')
msg = ab_cipher.decrypt(msg)
print(f'ALICE: Decrypted: {msg}')
```

```
    # Step 5
    print()
    print('========== Step 5 (A -> B (Nb - 1)Kab) ==========')
    input(f'ALICE: Press enter to send nonce - 1 to BOB... ')
    msg -= 1
    alice_bob_queue.put(ab_cipher.encrypt(msg))




def bob(alice_bob_queue, kb):
    b_cipher = Cipher(kb)

    # Step 3
    while alice_bob_queue.empty():
        sleep(1)

    print()
    msg = alice_bob_queue.get()
    print(f'BOB: Received keys from Alice {msg}')
    input(f'BOB: Press enter to decrypt... ')

    msg = b_cipher.decrypt(msg)
    print(f'BOB: Decrypted: {msg}')
    print(f'BOB: Kab: {msg[0]}')
```

```python
kab = msg[0]

ab_cipher = Cipher(kab)


# Step 4

print()

print('========== Step 4 (B -> A (Nb)Kab) ==========')

Nb = randint(1000, 9999)

print(f'BOB: Nonce: {Nb}')

input(f'BOB: Press enter to send encrypted nonce to Alice... ')

alice_bob_queue.put(ab_cipher.encrypt(Nb))

sleep(3)


# Step 5

while alice_bob_queue.empty():

    sleep(1)


msg = alice_bob_queue.get()

print()

print(f'BOB: Received encrypted message from Alice {msg}')

input(f'BOB: Press enter to decode... ')

msg = ab_cipher.decrypt(msg)

print(f'BOB: Decrypted: {msg}')

print(f'BOB: Nb: {msg}')

print(f'BOB: Key is Verified')
```

```
kdc_alice_queue = Queue()

alice_bob_queue = Queue()



kdc_thread = Thread(target = kdc, args = (kdc_alice_queue, ka, kb))

alice_thread = Thread(target = alice, args = (kdc_alice_queue,
alice_bob_queue, ka))

bob_thread = Thread(target = bob, args = (alice_bob_queue, kb))



kdc_thread.start()

alice_thread.start()

bob_thread.start()



kdc_thread.join()

alice_thread.join()

bob_thread.join()

print()

print('ALICE: Shutting down Alice server...')

print('KDC: Shutting down KDC server...')

print('BOB: Shutting down Bob server...')
```

```
PS F:\code\github.com\godcrampy\college-notes\nss\needham> python .\kdc.py
KDC: Starting KDC server ...
KDC: Waiting for Alice ...
ALICE: Starting Alice server (id: 100) ...

========== Step 1 (A → KDC (A, B, Na)) ==========
ALICE: Executing Step 1: Press enter to continue ...
ALICE: Nonce: 7614
ALICE: Done
KDC: Received from Alice Server [100, 200, 7614]

========== Step 2 KDC → A (Na, Kab, B, (Kab, A)Kb)Ka ==========
KDC: Executing Step 2: Press enter to continue ...
KDC: Payload [7614, 'qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=', 200, 'gAAAAABiWoKFU8NK
zKhJRyE1aflXRoKwEprR1Cz2rMKUqovlnlsV-4IOEeaI_DajYeA8PBaYllzSyCz14OVUSOKybUkmt6zjP2KBsVRV7Z
WF1hftLJLPtcoeI37uD-tWvgWnoC5SXkvC3oYgwoiF-PP_1UkMTtTEhQ==']
KDC: Kab: qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=
KDC: Encrypted Payload gAAAAABiWoKFySIso_ooc2rB3HN_7Ps_Hb_x67fQirwwPhJJFWwgftF-y1K8-nSo_es
RtwYIr4SyZ4BbkdX_Rz-kYLDjS92tJvnDHU9eWfhyanvMGE8nCuCqky-HUbAqCBl7vUjEYAhWfvyrinsS2p3Zmd06S
7CkMRQcfjQu864h4FbJK5hyeZJzDGiay0ADeVzSAV4OUdjl8kAyOHdpyuNgpnDoPJOgiLvCH78ygdcvuk3PRuSBCBl
rJq8HlzzVQEHkK6yJCQ_6q0bHuipg-zmZWaUKMM1oneF0EK1-8v1MD1U_HaPqa7au1reL166RxsiRpPOEnkEro5×1d
2lRUS2tgE2ZRqXEqKcV6a0OvNdMX8cI0qCQQkoUeButc_rILkgKH0LjfPTI
KDC: Press Enter to send payload to Alice ...
```

```
ALICE: Received encrypted message from KDC gAAAAABiWoKFySIso_ooc2rB3HN_7Ps_Hb_x67fQirwwPhJ
JFWwgftF-y1K8-nSo_esRtwYIr4SyZ4BbkdX_Rz-kYLDjS92tJvnDHU9eWfhyanvMGE8nCuCqky-HUbAqCBl7vUjEY
AhWfvyrinsS2p3Zmd06S7CkMRQcfjQu864h4FbJK5hyeZJzDGiay0ADeVzSAV4OUdjl8kAyOHdpyuNgpnDoPJOgiLv
CH78ygdcvuk3PRuSBCBlrJq8HlzzVQEHkK6yJCQ_6q0bHuipg-zmZWaUKMM1oneF0EK1-8v1MD1U_HaPqa7au1reL1
66RxsiRpPOEnkEro5×1d2lRUS2tgE2ZRqXEqKcV6a0OvNdMX8cI0qCQQkoUeButc_rILkgKH0LjfPTI
ALICE: Press enter to decode ...
ALICE: Decrypted: [7614, 'qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=', 200, 'gAAAAABiWoK
FU8NKzKhJRyE1aflXRoKwEprR1Cz2rMKUqovlnlsV-4IOEeaI_DajYeA8PBaYllzSyCz14OVUSOKybUkmt6zjP2KBs
VRV7ZWF1hftLJLPtcoeI37uD-tWvgWnoC5SXkvC3oYgwoiF-PP_1UkMTtTEhQ==']
ALICE: Kab: qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=

========== Step 3 (A → B (Kab, A)Kb) ==========
Alice: Press enter to send encrypted keys to Bob ...

BOB: Received keys from Alice gAAAAABiWoKFU8NKzKhJRyE1aflXRoKwEprR1Cz2rMKUqovlnlsV-4IOEeaI
_DajYeA8PBaYllzSyCz14OVUSOKybUkmt6zjP2KBsVRV7ZWF1hftLJLPtcoeI37uD-tWvgWnoC5SXkvC3oYgwoiF-P
P_1UkMTtTEhQ==
BOB: Press enter to decrypt ...
BOB: Decrypted: ['qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=', 100]
BOB: Kab: qsRK4pwpKQo8BOeE8Dr841HEkmYuZfD_zOav8VBqsQg=
```

```
========== Step 4 (B → A (Nb)Kab) ==========
BOB: Nonce: 9685
BOB: Press enter to send encrypted nonce to Alice ...

ALICE: Received encrypted message from Bob gAAAAABiWoKiWl5mANkdeXyQBJObRcPg1aTYjJppAM17MM1
dBMixiXRe7kr0Ne3eA6EWx5gn4f7Nti38n8Gctd3xgATuio5wDA==
ALICE: Press enter to decode ...
ALICE: Decrypted: 9685

========== Step 5 (A → B (Nb - 1)Kab) ==========
ALICE: Press enter to send nonce - 1 to BOB ...

BOB: Received encrypted message from Alice gAAAAABiWoKmh3ug-sDr07TWsJto7Bzj4t9vAFbflev3z6h
tGa1kTmif6IGQDcoinI75BF_Gs-sSKTUyQPbZ_w1MT65KWqCQeQ==
BOB: Press enter to decode ...
BOB: Decrypted: 9684
BOB: Nb: 9684
BOB: Key is Verified

ALICE: Shutting down Alice server ...
KDC: Shutting down KDC server ...
BOB: Shutting down Bob server ...
PS F:\code\github.com\godcrampy\college-notes\nss\needham>
```