Hanum Lee
30010205

Assignment 2

Test Cases

My program can't take usernames with spaces.

My program is not case sensitive for checking whether a word is in dictionary or not.

Assuming that database gets resetted every time.

1. Console:
   enroll.py test1 1a2
   auth.py test 1a2
   Output:
   Accept
   Access Granted
   Checking:
   In valid cases, the program should enroll and authenticate correctly
2. Console:
   enroll.py test1 1a2
   auth.py test1 122
   Output:
   Accept
   Not correct password. Access Denied
   Checking:
   Enrolled correctly, but authenticating wrong
3. Console:
   enroll.py test1 a
   Output:
   Invalid Password. Rejected
   Checking:
   The enroll program rejects password that is just word in dictionary.
4. Console:
   enroll.py test1 a1
   Output:
   Invalid Password. Rejected
   Checking:
   The enroll program rejects password that is just word in dictionary followed by number.
5. Console:
   enroll.py test1 1a

Output:
Invalid Password. Rejected
Checking:
The enroll program rejects password that is just number followed by word in dictionary.

6. Console:
   enroll.py test1 1a2
   enroll.py test1 2a1
   Output:
   Accept
   Invalid ID. Rejected
   Checking:
   If there is same ID already saved in database, program rejects
7. Console:
   enroll.py test1 1a2
   auth.py test2 1a2
   Output:
   Accept
   Not valid ID. Access Denied
   Checking:
   If there isn't ID in database that the user is authenticating, then auth.py should reject.
8. Console:
   enroll.py test 1234
   Output:
   Invalid Password. Rejected
   Checking:
   The enroll program rejects password that is just numbers.
9. Console:
   enroll.py test atab
   Output:
   Accept
   Checking:
   The enroll program should work with the password that is just word but not in dictionary.
10. Console:
    enroll.py test Ab12
    auth.py test ab12
    Output:
    Accept
    Not correct password. Access Denied.
    Checking:
    Case sensitive for password

Question 2:

1. The long term secret in Kerberos is the long term key for client, server and service.
2. The short term secret is the session key for client and server or service to authenticate. Key distribution center gives client session key which is derived from the long term keys between server and client.
3. If long term secret is leaked, it means that either the client's long term key or server's long term key has been leaked. The adversary who has recorded all of the previous network traffic would get new information such as the shared secret between the client and server. Also adversary would know the content of each ticket between the client and server.
   If short term secret is leaked, then adversary would know what job has been requested by the client to the service.
4. If the adversary has recorded all the transmission between the client and server, when long term key of client or server is compromised, then all the information transferred can be decrypted using either the client's key or server's key. Since adversary can decrypt previous messages transferred between server and client, it does not have forward secrecy when long term key for either client and server is compromised. If the session key between client and service is compromised, then adversary can look at all the job requested by the client to service by decrypting it with one of key.
5. (idea from:https://www.iasj.net/iasj?func=fulltext&aId=24444)
   We can implement Diffie-Hellman algorithm to achieve forward secrecy with Kerberos. Everything is same as original Kerberos until the authenticating between client and service using the ticket granted from ticket granting server. Client sends same thing as Kerberos but add g^a in the message. Service authenticate by sending the reply as Kerberos but add g^b. They compute g^ab, then use that as the session key. This will provide forward secrecy since we can change a and b everytime and it is hard for adversary to find g^ab for previous session.