Nolan Pofi

# ReadMe

My overall program is split up into three files Alice.py, Bob.py, and KDC.py and all three rely on my DES from homework one to do any encryption or decryption. Alice represents user A in the Needham-Schroeder protocol, Bob is user B, and KDC acts as the key distribution center. The files communicate with Alice acting as a server for both Bob and KDC, KDC as a server for Bob and client to Alice, and Bob is a client to both Alice and KDC. The program starts with Alice and Bob generating keys KA and KB respectively with the KDC through Computational Diffie-Hellman key exchange protocol the nature of which I will delve into later on. After all files know their respective keys Alice contacts KDC with the IDs of Alice and Bob. The KDC then proceeds to generate a new key KS for the current session between Alice and Bob. The KDC then creates a message to send to Alice which is encrypted using KA. The message consists of KS,IDA,T,EKB(KS,IDB,T) with IDA and IDB being their IDs, KS the session key, T which is the timestamp for when the KS was created by the KDC, and EKB stands for the fact the last part is encrypted with KB which Alice does not have access to. The timestamp is added to help prevent replay attacks it does this by Alice and Bob comparing the value of T to the current time and if the difference is greater than a day then neither will continue with the protocol preventing an attacker from resending a message older than a day. The whole message is encrypted under KA and sent to Alice. Alice then decrypts this message taking KS and checking for a valid T. The second half of the message Alice sends to Bob which promptly decrypts the message getting the session key KS and also verifying the timestamp T. Bob then encrypts a nonce using KS and sends it to Alice. Alice proceeds to decrypt the nonce and put it through function f. f is a simple function that just modifies the nonce. The modified nonce is then

encrypted with KS and sent to Bob. Bob decrypts this and proceeds to check if the received value matches the value Bob gets when it puts the original nonce through its f function which is the same one in Alice. If it does, then the key assignment has worked and Alice and Bob can continue sending messages using KS however my program simply ends with Bob outputting a True if the operation was successful.

My Diffie-Hellman key transfer is set up in the following way. All three files have the same q and α with q = 353 and α = 3. These values were taken from an example in the slides just to test that it worked properly and I asked in office hours what a better q would be and was told this was fine to just leave. I then assigned each one with their private key x Alice x = 19, Bob x = 55, and KDC x = 255. Each one then generates its own public key Y through the equation $Y = \alpha^x \bmod q$. To generate the shared key between Alice and KDC and Bob and KDC they send their public keys to each other $Y_b$ and will compute a key that is the same by doing $Y_b^{xa} \bmod q$. With xa representing the secret key and $Y_b$ as the others public key. Since $Y_b = \alpha^{xb} \bmod q$ by the new equation gives $(\alpha^{xb})^{xa} \bmod q$ which is equivalent to $(\alpha^{xa})^{xb} \bmod q$ which is what the other file has allowing both to have the same key which an attacker can't get as they'd have to find discrete logarithms.