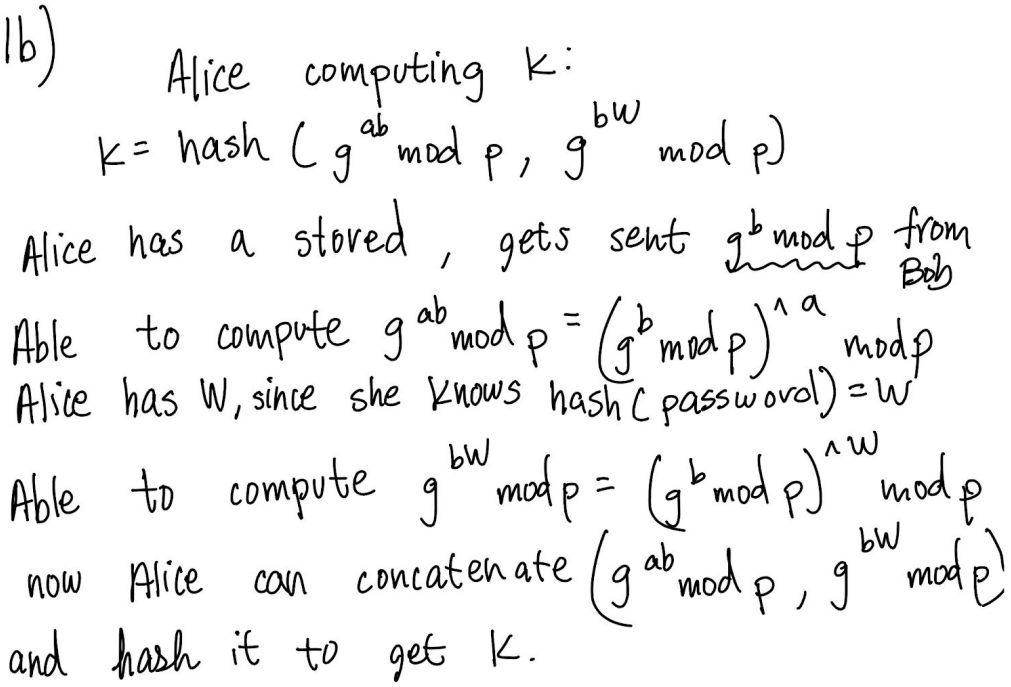
Homework 5

Jake Maschoff
U0581770

1. Question 1
    a. Chapter 12, Problem 5, page 302
        i. **Bob keeps two values of n. A value of n that he has stored in his database, and an n that he sends out, that we will call N. Initially N = n - 1, but the N that he sends out is decremented each time Bob sends out an N. For each connection, Bob stores n - N (where N is the one that was just sent out), which will be the number of hashes you need to compute on what the authenticator sends to Bob for that current connection. When an authenticator sends the correct value of hash(password)$^{N-1}$(where N is the correct N for that session), Bob updates his n = N - 1, and the hash value to what the authenticator sent Bob in that current connection. That way if Bob does shut down before Alice can respond and when Trudy opens a connection with Bob, Bob will ask with a decremented N of what was asked Alice. And since Trudy can't undo the hash once, Trudy will not be able to respond with the correct value and the value sniffed from Alice will be incorrect. Once N or n hits 1, a new password must be made between the user and Bob.**
    b. Chapter 12, Problem 15, pages 302-303.
        i. Calculating K for Alice and Bob



Bob computing K:

$K = \text{hash}(g^{ab} \bmod p, g^{bW} \bmod p)$

Bob has b stored, gets sent $g^a \bmod p$ from Alice

Able to compute $g^{ab} \bmod p = (g^a \bmod p)^{\wedge} b \bmod p$

Bob has W stored as prior knowledge.

Able to compute $g^{bW} \bmod p = (g^b \bmod p)^{\wedge} W \bmod p$

now Bob can concatenate $(g^{ab} \bmod p, g^{bW} \bmod p)$

and hash it to get K.

ii. If someone is impersonating Bob, they are able to perform an offline dictionary attack. That is because they are able to compute $g^{ab} \bmod p$, since they are making b and $g^a \bmod p$ is sent from Alice, and the other half of the key, $g^{Wb} \bmod p$ is already half-known because $g^b \bmod p$ is made by the impersonator. That means the only unknown is W. Since the impersonator is sending the challenge c_1, and the value is sent back from Alice encrypted with $K\{c_1\}$, the impersonator can change W until they get the same value as $K\{c_1\}$ sent from Alice using the key they are fabricating against the original challenge c_1 that they sent.

2. Question 2
    a. Chapter 16, problem 1 (each part – 1 point).
        i. 16-2
            1. Has:
                a. PFS
                    i. Uses sessions key
                b. Escrow passive protection
                    i. Uses session key
                c. Escrow active protection
                    i. Uses signature keys that are not escrowed
            2. Does not have:
                a. Identity Hiding

        i. Able to ‘Alice’ and ‘Bob’
    b. PFS Identity Hiding
        i. Able to ‘Alice’ and ‘Bob’

ii. 16-2-1
1. Has:
    a. PFS
        i. Uses session key
    b. Escrow passive protection
        i. Uses session key
    c. Identity hiding
        i. Identity hidden behind public-key encryption
2. Does not have:
    a. Escrow private protection
        i. Able to create session keys between Alice and Bob since we are able to decrypt the messages using the escrowed key and re-encrypt the value using the key as well
    b. PFS identity hiding
        i. Once the private key is found, able to decrypt the values and see ‘Alice’ and ‘Bob’

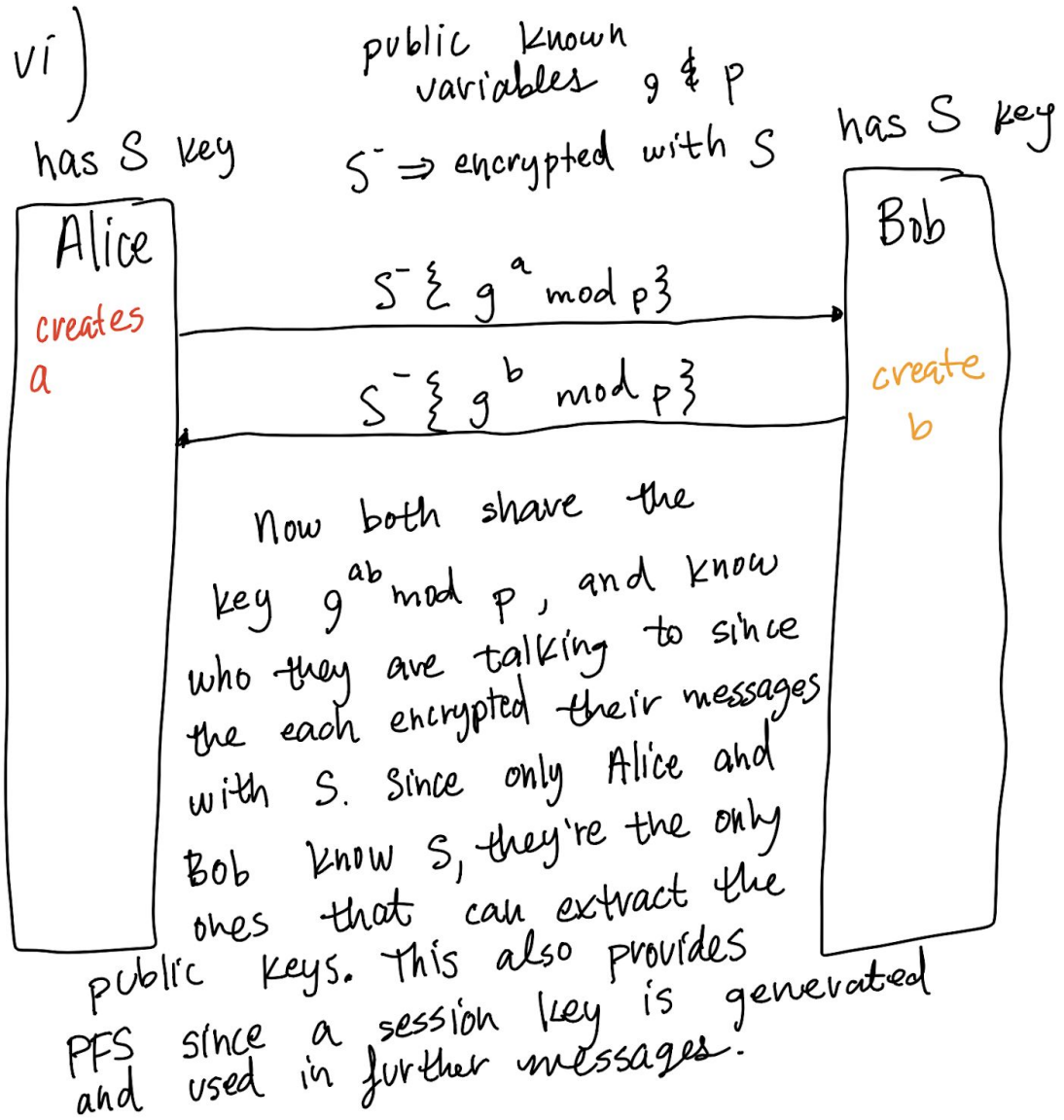
iii. 16-4
1. Has:
    a. PFS
        i. Uses session key
    b. Escrow passive protection
        i. Uses session key
    c. Escrow active protection
        i. Uses signature keys that are not escrowed
2. Does not have:
    a. Identity hiding
        i. Active attacker will be able to identify ‘Alice’
    b. PFS identity hiding
        i. Active attacker will be able to identify ‘Alice’

iv. 16-9
1. Has:
    a. PFS
        i. Uses session key
    b. Escrow passive protection
        i. Uses session key
2. Does not have:
    a. Escrow active protection

            i. With the known S, an active attacker can create session keys between Alice and Bob, man-in-the-middle attack
        b. Identity hiding
            i. ‘Alice’ and ‘Bob’ are known to everyone
        c. PFS Identity hiding
            i. ‘Alice’ and ‘Bob’ are known to everyone

v. Nonce protocol
    1. Has:
        a. Identity hiding
            i. No identity information sent
        b. PFS identity hiding
            i. Not identity information sent
    2. Does not have:
        a. PFS
            i. Once keys are broken, they are able to get the nonces and create the key to decrypt all past messages
        b. Escrow passive protection
            i. With known keys, you are able to decrypt the values to see nonces and create the key
        c. Escrow active protection
            i. With known keys, you are able to decrypt the values to see nonces and create the key

vi. Assume Alice and Bob share a secret S. Design a protocol in which they can do mutual authentication and establish a shared secret with PFS. Can it be done without Diffie-Hellman or any other form of public-key cryptography?



1. No it can't be down without any other form of public-key cryptography because the only other form would be using symmetric or asymmetric keys, and those don't provide PFS due to the ability of someone finding the secret keys and being able to decrypt past messages.

vii. 16-2-2

1. Has:
    a. Escrow passive protection
    b. Escrow active protection

i. Still generates a session on the basis of the seed that is a shared secret. Hard to tell by the question if any other form of encryption is being used on top of this, like hashing. But I am assuming that even with S known, it will be hard to generate the next upcoming session key that is created by the random generator because of what sequence the random number generator is on.

2. Does not have:
    a. PFS
        i. If someone figures out the seeds stored on the server, they would be able to have the value created on the client-side (a), and server-sider (b). Thus being able to create the session key and look at past messages sent between the parties.
    b. Identity hiding
        i. 'Alice' and 'Bob' are seen to everyone
    c. PFS identity hiding
        i. 'Alice' and 'Bob' are seen to everyone

3. Question 3
    a. Let H be a hash function that is both hiding and puzzle-friendly. Consider G(z) = H(z) ‖ zlast, where zlast represents the last bit of z. Show that G is puzzle-friendly but not hiding.
        i. **It is puzzle-friendly because it still uses H(z), which is a puzzle-friendly hashing algorithm, thus making it infeasible to compute G(z) due to the hashing function.**
        ii. **It is not hiding since the last bit is concatenated onto the end of G(z), thus the user would be able to know the last bits of what was inputted. If the value that was inputted was just 1-bit, they would know the value!**
    b. In ScroogeCoin, suppose Mallory by chance generates a public, private key pair that matches another person's public, private key pair. Could she create valid transactions stealing coins belonging to the other person? Explain briefly.
        i. **A transaction is valid if coins are created in a previous transaction (which we will assume that this person with the same key has obtained/created coins in a previous transaction), the coins have not already been consumed by some previous transaction (which we will assume this person with the same keys has not spent these coins yet), the total coins consumed and created are equal (which would be true since Mallory would ensure that this is true), and the transaction is signed by all owners of the coins consumed. Since Mallory would have the same private key pair, they would be able to**

sign off on this transaction, thus using the coins generated by the other person. So yes, Mallory would be able to create valid transactions stealing coins belonging to the other person.

c. Even when all nodes are honest, blocks will occasionally get orphaned or won't be included in the longest chain: if two miners Minnie and Mynie discover blocks nearly simultaneously, neither will have time to hear about the other's block before broadcasting hers.
    i. What determines whose block will end up on the consensus branch?
        1. **It's random since in each round the node gets to broadcast its block next is chosen at random. However, this randomness is incentivized via transaction fees for other nodes to include this block in their block.**
    ii. What factors affect the rate of orphan blocks?
        1. **<u>Transmission time</u> between nodes, so if nodes don't hear about completed nodes for a long time, otherwise might start computing for blocks that were accepted by a majority awhile ago. Also how new blocks are <u>queued</u> in the mining hardware could have an effect. If they are queued on something other than arrival time, this could increase the number of blocks that are passed up. The time to <u>process</u> a correct block also increases the rate as some blocks could get delayed until they are validated thus making them orphaned if another block is picked or created in the meantime.**
    iii. If Mynie hears about Minnie's block just before she's about to discover hers, does that mean she wasted her effort?
        1. **It depends. If Mynie's block isn't valid then it obviously won't be adopted and Minnie's block then will be. Also, people could take a while to accept Minnie's block so that would allow Mynie to complete theirs and then Mynie could add an incentive for everyone to accept her block quicker with a transaction incentive. Also, Mynie could send her block quicker to a majority of nodes before Minnie's block propagated through the network if the nodes they sent their block too were slow. If Minnie's block gets adopted into the chain before Mynie's, though, then her effort was wasted.**

4. Question 4
    a. PROGRAMMING ASSIGNMENT (turned in)
5. Question 5
    a. What are the advantages of building security at the TCP layer?
        i. **Provides the authentication of TCP headers. This helps provide security against DDoS attacks by not allowing for any TCP packet injection techniques to be used.**
    b. What flexibility does TCPsec provide for encrypting the TCP segment?

    i. **The TCP header and payload can be encrypted on a connection basis. Depending on what connection is being run, or who it's with, you are given the option to actually encrypt these values or not. TCPsec provides end-to-end security while also making the protocol firewall friendly, in comparison to IPsec. TPsec is also backwards compatible with TCP which provides a level of flexibility.**

c. How does TCPsec interoperate with NATs?
    i. **In the handshake phase of TPsec, you can use a NAT option. With this option set the client sends the open ports and source IP addresses used for the NAT. The TCPsec server can then compare against these values when they receive packets on certain ports and modify the packets sent back. The TCPsec server also responds to this handshake message with a NAT option if it detects a NAT with its open ports and source IP addresses, so the client can make adjustments as well.**

d. Why does TCPsec handshake perform better than SSL?
    i. **SSL's handshake performs worse because it must perform TCP's 3-way handshake on top of SSL's handshake process. Whereas with TCPsec handshake is just a 4-way TCP handshake, making it much faster.**