Homework 4: Internet Security

1. **Authentication protocols:**

   (a) Having central sign-on facility improve security in many ways, given below:

   - User has to remember only one password, using that they can generate their token and this token allow them to access all campus's sites. If user has multiple credentials each for a site then there is higher chances of attack.
   - In central sign-on facility, we have one interface for login, university can enforce any security rules by changing only one interface.

   (b) It can hurt security in multiple ways :

   - Using a single password increases the chances of password vulnerability.
   - When Sign-on facility fails or down, access to all related systems is lost.
   - Having one node to for authentication can increase the risk of malicious attacks.

   (c) After authentication, user is redirected back to site A with u, username and $Sign(u)$, digital signature. This username and digital signature is used to authenticate all other university sites. If site A is controlled by an attacker, attacker has access to both username and digital signature of the user. Attacker can easily access all other campus sites using the same username and signature as a legitimate user.

   (d) The digital signature received by the central sign-on facility should be one time use only. Every time user try to access a campus site, he needs to generate new signature. This would fix out problem being identified in last part and strengthen the security protocol.

   (e) Mallory can perform man in the middle attack or spoofing attack and just replay the messages from one to other node of the group and can easily read the information of the messages.

When someone start the conversation in group let's say Alice, Alice send $n1$ to bob. Mallory can read the message in middle and replay this to bob. Then bob calculate $HMAC_k(n1||Alice)$ and send back this HMAC value and $n2$.

Again mallory in read the message comes from bob and replay this message to Alice. Alice verify the HMAC value and this value should be correct because it comes from bob who has the shared key and send $HMAC_k(n2||Bob)$ value back to bob. Mallory again replay this message to bob. Bob verifies the HMAC comes from Alice via mallory and it will be correct because alice also has the shared key. Alice and Bob thinks that mallory is also a part of the research group and she can easily access all the information in between both.

(f) The protocol can be enhanced by using format and timestamp or time to expire field associated with the message. Having timestamp with the message provide extra security to a user. There are following ways to do it:

- One way to do mutual authentication process that exchanges user messages may be implemented as follows
    1. Alice sends an encrypted message to Bob to show that Alice is a valid user.
    2. Bob verifies message:
        a. Bob checks the format and timestamp. If either is incorrect or invalid, the session is aborted.
        b. The message is then decrypted with Bob's secret key and Alice's message.
            i. Bob checks if the message matches a valid user. If not, the session is aborted.
    3. Bob sends Alice a message back to show that Bob is a valid user.
    4. Alice verifies the message:
        a. Alice checks the format and timestamp. If either is incorrect or invalid, the session is aborted.

b. Then, the message is decrypted with Alice's secret key and Bob's message.

    i. Alice checks if the message matches a valid user. If not, the session is aborted.

5. Now, both parties have been confirmed as being who they say they are and are safe to communicate with.

- Other way to do it by time to expire: We attach time to expire field with each message considering how much time it will take to travel to other user and drop the packet after time is up. So, now Alice sends a message to bob and mallory is trying to read and replay the message, time to expire will reach its limit before reaching to bob and session will be aborted because no one is able to verify anyone else.

  We also has a timestamp field associated with message which tells users about the validity of the message. If somehow bob is able to get the message before time to expire ends, bob will abort the message just by looking timestamp of the message. By using these both field we can restrict mallory to read the information and try to be a part of the group. This is how by updating the protocol will allow us to make this mutual authentication securely.

2. **Password cracking:**

(a) There are total 62 characters (26 from a-z, 26 from A-Z and 10 from 0-9) and password is 8 characters long random password. So, possible combination of password would be 62^8.

Given that the passwords are randomly chosen and attacker can carry out 4 million hashes per second. On average, to crack a password using brute force will take $62^8/2(4million)$ seconds which is roughly equal to the 7581 hours.

(b) Because it takes 7581 hours on average to crack a single hash, the attacker will require a 7581 node botnet to crack one hash each hour on average.

(c) The table would have 62^8 entries (according to part 1), each entry occupy 8 bytes for
the password plus 32 bytes for the hash. So, the table occupy total 40 byte per entry and we have 62^8 entries. Total table size would be $62^8 * 40Bytes$ which is roughly equal to the 7.76 pebibytes.

(d) The table has a start and an end, allowing us to obtain all of the passwords in the chain simply by knowing the starting and end points. So, We just store the start and end of each chain which require 16 bytes. Thus, the equation for the number of bytes in the table in terms of the chain length k and the size of the password set N, is $16\lceil N/K \rceil Bytes$.

(e) If K=5000, by using formula, $16 * \lceil 62^8/5000 \rceil == 698,688,337,872 bytes$ which roughly equal to the 651 Gibibytes.

(f) The attacker first need to compute the entire table and it will take twice the amount time for each password. This means that we will need $4 * 7581 hours$ to calculate the passwords which gives us 30324 hours. So, it takes roughly 30324 hours for an attacker to create the table if the attacker can add 2 million chain elements per seconds.

(g) It takes 4 times as long (or as big a botnet) to build the table using rainbow tables, but it occupies only about 1/12500 as much space.

From part (a), we can say that it takes approximately 4 times as long to create the table as it takes 7581 hours in part (a) and it takes 30324 hours which is 4 times the time taken in part (a).

If we talk about size, It takes as long as the size of botnet in this case 7851 when compared to the apart (b) to build the table using rainbow tables.

In part (c) it takes 7.75 pebibytes of memory and the rainbow table just takes 651 Gibibytes which is 1/12500 the space.

(h) We now have a password as well as a secret key. As a result, the attacker's job becomes more difficult. Previously, we just kept a pre-computed rainbow table that could be used against any site.
The attacker must now generate a distinct table for each site after adding the server secret key. This raises the attacker's difficulties, since creating separate tables for each site requires a lot more storage and computational resources. This helps us to fight against rainbow attacks in part.

(I) There are the following designs that would increase the protection :

- Instead of using server secret values for each password, we can use user-generated secret values which increases the security of the system. As a result, the attacker will have to construct tables for each user, which will be more complex and time-consuming, blocking the attacker from attacking the system.
- If we can use multiple iterations of hash, the attack has to use more computation power and it will take longer to crack a password and slow down the attacker. So doing many iterations strengthen the security and prevent attacks.

(j) By using bitcoin mining hardware attacker can generate $180 * 10^9$ hashes per second. So, on average brute force attack will take $62^8/180 * 10^9 seconds = 606 seconds$ which is roughly equal to the 10 minutes. This means by using that kind of hardware, attacker is able to break SHA-256 within a few minutes.