ASSIGNMENT-4

1)

A)

- Passwords can be very tricky to remember and having to come up for passwords that are very difficult to crack can be really difficult for the users. Easy guessed passwords can be easily broken by the malicious users and can easily compromise the security of the campus. Single sign on facility can help to overcome this problem, because the user need not come up with such difficult passwords and even remember such difficult passwords. Using single sign on facility allows the users to just provide the username and a password to such a facility and authenticate themselves.
- This also allows the user to have access to entire suite of campus resources from a single portal and with just one pair of credentials instead of a dozen of passwords. This also means we reduce the number of chances the attacker can attack the user's dozens of credentials because most of the time people end up using the same password for multiple accounts. This means that if the adversary compromises one password, adversary can gain access to all the accounts using the same password.

B)

- One problem can be if the central sign-on facility is the single point of failure. If this facility goes down then the researchers might not be able to log-in in any of the services and this will cause a lot of problems in the research work done by the university.
- This also means if this facility is compromised and has been attacked by the adversary, then the user's data is at risk and the adversary can impersonate any user.

C)

- Suppose the user provides some information to the sign-on protocol for its authentication, then the server authenticates the user and sends the username (u) and digital signature (sign(u)).
- The website can then just check the signature to make sure the user is authorized to visit the website.
- If suppose site A is controlled malicious user(attacker), the user when visits the website, authenticates itself using the sign-on facility.
- The facility then sends back username(u) and digital signature (sign(u)).
- If the attacker website can store the username(u) and digital signature it receives from the sign-on facility, then attacker can use this username(u) and digital signature of the legitimate user and try to access some other site B and since this user is legitimate, the digital signature(sign(u)) and username(u) are also valid, all websites will grant access to the attacker assuming that it was a legitimate user.

- This is how the attacker can impersonate a user and get access to all the other websites.

D)

- A simple change to the protocol could be that instead of returning digital signatures that are valid all the time, we can return digital signatures that are valid only for one time.
- Each a time user authenticates itself, a different digital signature needs to be generated for that user.
- This helps to strengthen the security because then the attacker cannot just username and digital signature it received, because if the attacker does the signature is valid only for one time and the attacker will not be able to access any other websites by impersonating a valid user's credentials.

E)

- To make Bob think Mallory is a part of a group we are going to take advantage of the replaying the messages.
- The question also mentions that someone might try to start a conversation with Mallory. So, Mallory will do something like a spoofing attack to make Bob think Mallory is a part of the research group.
- If Alice tries to initiate a conversation by sending the necessary information, Mallory can then just replay this message to Bob.
- Seeing this message Bob will try to send the calculated hmac on (n1|| Alice), a random nonce n2 and Bob to Mallory.
- Mallory will then use this message and forward it back to Alice.
- Since the hmac are calculated by Bob who shares the secret key, Alice will verify that this Hmac is correct and will in turn send a meesage containing the Hmac on n2||Bob and Mallory can just replay this message to Bob.
- The hmac which was just replayed by Mallory is correct because it comes from Alice who also knows the secret key.
- This will be verified by Bob and since it is correct, Bob will think that Mallory is a part of the research group and Mallory can then get herself all the information shared between the members of the research group.

F)

- We can modify the protocol by adding one more parameter like a timestamp or a time to live field in the messages, this ensure that no message can be played back to other users.
- We can set the time to live field to round trip time it takes for a message from one user to another user, this also allows to solve the problem of not dropping messages from real users in case the network gets jammed up.

- We can decrease the time to live field and if the time to live field in a message is 0, the receiver will just drop the message packet.
- So, in case Alice sends a message to Mallory trying to initiate a conversation and if Mallory will just replay the message to Bob, the time to live field will still keep on decreasing and by the time, the message reaches Bob, the time to live field would have been 0.
- And since we defined that if a receiver gets a message with time to live field as 0, it will drop the message.
- So, Bob will drop the message and thus Mallory will never will be able to prove to Bob that she is a legitimate member of the research group.
- Even if the time to live doesn't become 0 by the time it reaches a receiver, we also have timestamp field and we can have a field indicating till what time this message is valid. If that time+timestamp value passes the timestamp at which the messages arrive, the receiver will drop the message.
- This is an added security measure because this valid time period will be good for only one way communication because if Mallory tries to just replay the message, the timestamp by which the message reaches Bob, the valid time period would have passed and Bob will just drop the packet.
- This shows that the above updates to the protocol will allow us to make this mutual authentication securely without making some outside user prove that the user is a part of the research group.

2)

A)

- The problem states that we can use passwords that contain characters from a-z, A-Z and 0-9.
- There are 26 characters from a-z, 26 characters from A-Z and 10 characters to choose from 0-9. We can also repeat the characters in the password.
- This means that we have n = (26 + 26 + 10) ^ 8 possible passwords. This means that are in total 218,340,105,584,896 passwords possible.
- We know that the passwords are randomly chosen and all the unknown passwords have the equal probability of having any of the n values.
- Since we are supposed to perform a brute force attack on the passwords, and it is assumed that we can have generate 4 million SHA- hashes per second.
- On an average the number of guesses for a brute force can take n/2 attempts. Thus it takes n/(2(4 million))s. This means we have 218,340,105,584,896/(2(4million)) which gives us 7582 hours. (7581.2 hours, so I rounded it to 7582)

B)

- A botnet would help us to crack the passwords because this network of bots will follow the order of the master which in this case is to generate 4 million hashes per second.
- Since it takes us 7582 hours as shown above and the question asks us to crack the hash at an average rate of one hours.
- So, we will need a botnet of 7582 node to crack one hash per hour on average.

C)

- The question assumes that the adversary will compute the hash for every valid password and then store this hash, password pair in the table.
- Since we know there are 218,340,105,584,896 passwords possible and adversary will calculate all the hashes for every password.
- We will need 8 bytes to store every password and 32 bytes for the 256-bit hash we get for each password. This means that the entire entry for a single password-hash value takes 40 bytes.
- Since we have 218,340,105,584,896 passwords so the entire table will take 218,340,105,584,896 * 40 bytes which accounts to 8,733,604,223,395,840 bytes.
- Thus, the entire table will take 8,733,604,223,395,840 bytes.

D)

- The rainbow table has a start and end which allows us to get all the passwords in the chain by just knowing the start and end point of it. The start is to provide us with a starting point and reconstruct the chain while the end point is sufficient enough to recognize whether hash value is likely to be part of the chain. Also, it is given that there are no collisions in the rainbow table.
- Since we just store the start and end, we just require 16 bytes for this.
- 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[N/k] bytes. (We will take the ceil value for this equation).

E)

- If k = 5000 and using N= 218,340,105,584,896 and substituting it in the above equation we get (16*218,340,105,584,896)/5000 which gives us 698688337872 bytes.
- So, the rainbow table created by the attacker will take 698688337872 bytes which is much less compared to the above hash-password table that the attacker used.

F)

- We now know that the attacker can add 2 million chain elements per second. So, the attacker will first need to compute the entire table and it will take twice the amount time for each password.
- This means that we will need 218,340,105,584,896 / (2*10^6*3600) to calculate the passwords which gives us 30325 hours.

- So, it takes roughly 30325 hours for an attacker to create the table if the attacker can add 2 million chain elements per seconds.

G)

- From the answer of A, we can conclude that it takes approximately 4 times as long to create the table as it takes 7582 hours in the answer of A and it takes 30325 hours which is 4 times the time taken in A.
- It takes as long as the size of botnet in this case 7852 when compared to the answer in B to build the table using rainbow tables.
- In answer C it takes 8,733,604,223,395,840 bytes of memory and the rainbow table just takes 698688337872 bytes which is 1/12500 the space in answer C.

H)

- Earlier we were just storing the hash value we get by SHA-256(password) but now we have one more parameter along with the password we now use the server secret key appended to the password and then we calculate the SHA-256 value.
- So earlier we just maintained a standard precomputed rainbow table that worked against any site.
- Now after appending the server secret key, we make the task difficult because now the standard pre-computed rainbow doesn't help the attacker.
- If the attacker wants to still attack, he/she will have to compute a separate table for each site. This increases the difficulty of the attacker because making different tables for each site will require a lot more storage and more computing power. This allows us to partially defend against rainbow attacks.

I)

- One strategy would be to make the attacker spend more computing power if he wants to attack the SHA mechanism. We can do so by using several iterations of SHA.
- Doing so allows us to come up with a design like SHA (SHA (SHA(secret-key|| password))). This will take a large number of resources to attack such a design like above one. Because to brute force one SHA, it took us lots of time, brute forcing several such SHAs will take even more time and then making a table which is separate for several sites will even require more resources and more storage. So doing so allows us to strengthen the security and prevent the attacks from user.
- Another correction to the algorithm could be that instead of using server secret key for each password we can have a per user random data to increase the security of the system.
- So now the attacker will have to generate tables for each user which will be more tedious and require more computing and storage thus preventing the attacker to attack the system.

J)

- Now the attacker can generate hashes at the rate 180 GH/s which means 180*10^9 hashes per second.
- Brute forcing using the above parameter will now take 218,340,105,584,896/(2*180*10^9) which means 606 seconds and 0.16 hour (10 mins).

- This means that the attacker can use such a ASIC hardware and be able to break SHA-256 within a few minutes.