**Quiz 2 Report**

111550130 李慕庭

**Problem 1-(a)**

2FA is usually commonly based on (1) sending certain time-sensitive authenticated code by email or SMS (2) sending certain tokens to apps in phones or other devices (ex: Google Authentication) (3) Biometrics such as Fingerprint or FaceID.

2FA by SMS might be attacked by phone-based attacks (such as SIM swap scams), which can intercept the authenticated code. And for 2FA by email or sending tokens to apps, might also be attacked by Man-In-The-Middle attacks. Even if the token has a time period, hackers can conduct a replay attack in the period. Another issue is that (1) and (2) are time-based one-time passwords, which highly rely on a single-point server. If there is one one-point failure, it might lead to cybersecurity issues.

In terms of the risk of token interception and related attacks, first, strengthen network security (such as using VPN, avoid using public Wi-Fi …etc.). Second, to prevent replay attacks, we should make sure all tokens are one-time and shorten their time period. Besides, servers also need to protect tokens from being stolen properly, for example, build a tokens revocation mechanism that once stolen, invalidates authentication tokens and asks the client to request tokens again.

On the other hand, to avoid single-point failure leading to token loss, we can also use HMAC-based one-time passwords. Once a user registers on the server, the server will send a secret key to the user with a counter initialized to zero. When we need to authenticate, the user will calculate a password based on the counter and secret key and send it to the server. The server also calculates a password based on the same counter and the secret key and checks whether it matches the password sent from the user. Since users and servers calculate passwords locally and based on a counter, which prevents the risk of data loss during transmission.

**Problem 1-(b)**

If the Yangming and Guangfu campuses want to connect and share their research resources securely by VPN, we have to consider the following requirements. First, the security of connections between campuses should be highly ensured. Second, how to manage access rights for data and resources precisely and efficiently. Last but not least, how to support so many students and faculty members to transmit and access data simultaneously.

To fit the aforementioned requirements, I think a site-to-site VPN might be a suitable VPN architecture. Site-to-site VPN will build encrypted VPN channels between two campuses; therefore, resources and data can be transmitted through the VPN channel without being exposed to the public internet. Besides, in terms of easy management, servers on two campuses can easily manage access rights by blocking all access to resources that didn’t come from the VPN channel or internal systems. Finally, to afford larger network traffic, site-to-site VPNs can improve capacity by increasing the number of VPN gateways and the bandwidth of VPN channels.

To build a site-to-site VPN, we must configure several components. First, we need to choose a protocol. For example, the IPSec protocol is one of the most used protocols. Next, we need to set up VPN servers on both campuses to handle encryption/decryption of data and the creation of security channels. Besides, we also need to build an internal system to control access rights for different resources. For example, a role-based access control system, gives users one or multiple roles and assigns different permissions to each role (instead of each user). Moreover, we can improve the reliability of this site-to-site VPN by failover mechanisms or multiple VPN gateways to prevent data loss and single-point failure.

**一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。Problem 2**

For sha1 encryption, I used sha1 tool in **hashlib** python package.

For easy and medium hash, I enumerated all possible passwords in the given txt files one by one and checked if sha1(password) equals to hash. If identical, then we find the password.

For the leak hacker hash, first I tried out all possible salt in the given txt files to find salt such that sha1(salt) equals salt\_hash. After finding salt, I tried out all possible passwords in the given txt files and checked whether sha1(salt + password) equals hash value, if equivalent, then we find the password. In terms of attempts, including attempts to find salt and the password.

**一張含有 文字, 螢幕擷取畫面 的圖片

AI 產生的內容可能不正確。Problem 3**

Here, I used SHA-256 tools in **hashlib** python package.

First, compute preimage based on my studentID (111550130). i.e. preImage = sha256(“111550130”). Next, simulate mining process in each round. At the beginning of each round, I will check if first (# of rounds) digits of block hash value equals first (# of rounds) digits of studentID, then we can skip this round (without nonce); Otherwise, I enumerated all possible nonce from 0x00000000 incrementally. Once I found a possible nonce such that sha256(prev\_block\_hash + nonce)’s first (# of rounds) digts matched first (# of rounds) digits of studentID, record sha result as prev\_block\_hash , wrote logger and went to next round. If I didn’t find any possible nonce, I output error log and end the program.

Besides, to make sure that the mining process won’t be stopped or limited by computer efficiency, I ran my program on Google Colab. (Without changing the main program, only added commands of amount drive and copy log)