i) a) Identify the functions of PKI with single CA and Hierarchical PKI architecture

- A public key infrastructure (PKI) includes the CA's, the physical infrastructure and the formulation and enforcement of policies/ procedures.
- It also includes the entire gamut of services required in supporting the use of digital certificates.

→ Certificate creation, issuance, storage and archival
→ Key generation and key escrow
→ Certificate/Key updation
→ Certificate revocation

Functions:

→ It provides a safe archival facility for all issued certificates.

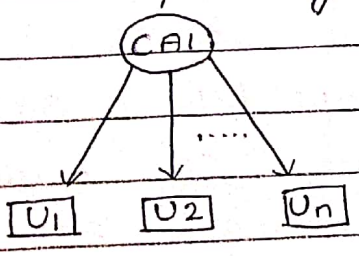
Architectures of PKI

i) PKI with single CA

CA1 could issue certificates to multiple users U_1, U_2 etc enabling any pair of these users to communicate securely using certificates exchanged between them. Each arc in the figure is a trust relationship

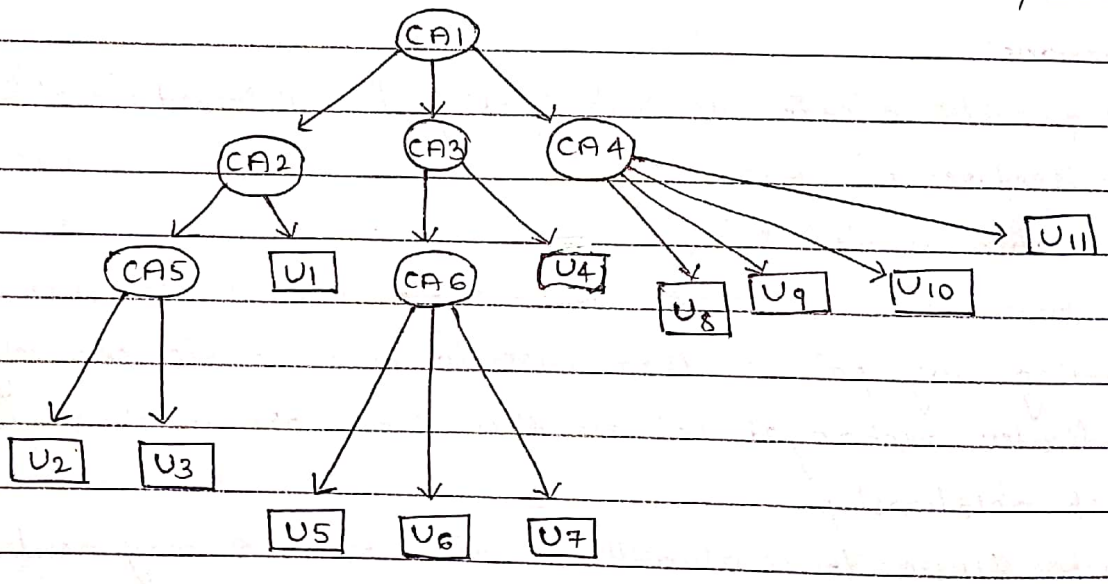
Suppose if these tens of millions of users who may need certificates. It is not practical for CA1 to issue certificates to them all especially if the type of certificate issued is the one that requires much background checking.

Eg: the arc from CA1 to U2 expresses the fact that the CA1 vouches for U2's public key in the certificate issued by CA1 to U2.



Hierarchical PKI architecture:

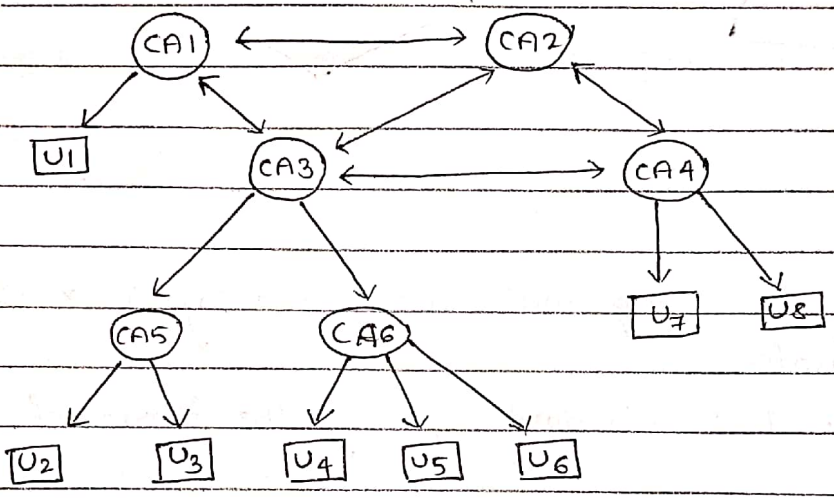
- A practical solution to the problem of scalability is to have CA1 certify other CA's who in turn certify other CAs and soon. This creates a tree of CAs known as hierarchical PKI architecture. Here, CA1 issues certificates to CA2, CA3 and CA4. CA2 in turn issues certificates to CA5 and end user U1. CA5 issues certificates to users U2 and U3. The advantage of this approach is easy scalability - each CA is responsible for certifying a limited no. of users or other CA's. CA1 is the root CA is sometimes referred to as the trust anchor. It is expected that every node in the tree will know the root CA's public key.



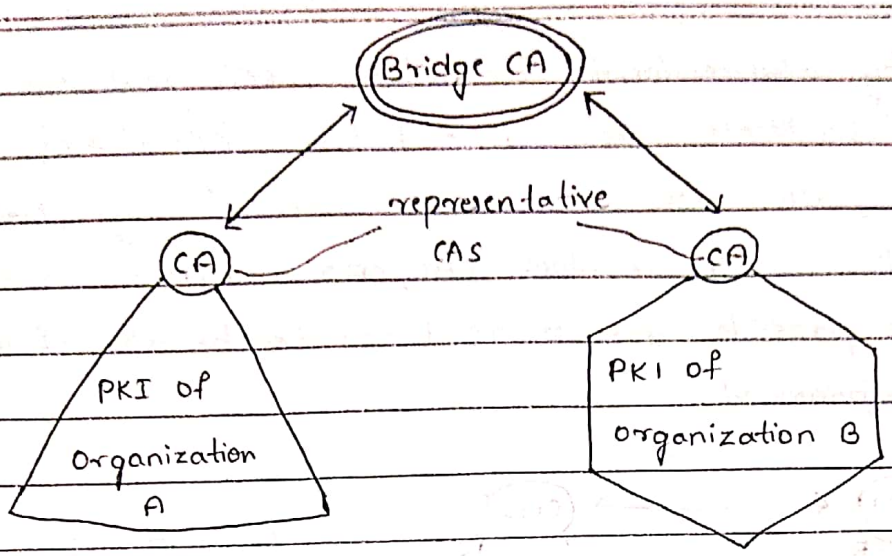
b) Identify the functions of Mesh based PKI architecture and bridge-based PKI

- In general, the trust relationships between CAs may be more numerous than in a simple tree of PKI. A more dense web of trust is referred as mesh-based PKI. This could include mutually trusting CAs - CA1 trusting CA2 and CA2 trusting

CA1 depicted by a bidirectional arc between CA1 and CA2
There may be multiple trust paths between CA1 and CA2.
Eg: One trust path users U1 and U7 passes through CA1, CA3 and CA4. Another trust path involves CA1, CA2 and CA4.
Multiple paths provide greater resilience in the even of one or more CAs being compromised.



- Bridge-based PKI, is motivated by the need for secure communications between organizations in a business partnership. A bridge CA is introduced that establishes a trust relationship with a representative CA from each organization. This is accomplished by the bridge CA and the organizational representatives issuing certificates to each other. The representative CA is one that has a trust path to all of the users in that organization. The below figure shows a bridge CA that extends the web of trust between two existing organizational PKI's.



c) List the fields along with their meaning in X.509 digital certificate.

X.509 is an ITU standard specify the format of public key certificates.

- Certificate Serial Number and Version.

Each certificate issued by a given CA will have a unique number.

- Issuer information.

The distinguished name of an entity includes his/her/its "common name", e-mail address, organization, country, etc

- subject information

This includes the distinguished name of the certificate's subject or owner.

- Subjects public key information.

The public key, the public key algorithm, and the public key parameters

- Validity period

These are two date fields that specify the start date and end date between which the certificate is valid.

- Certificate signature and associated signing algorithm

information.

It is necessary to verify the authenticity of the certificate. For this purpose, it is signed by the issuer. So, the certificate should include the issuer's digital signature and also the algorithm used for signing the certificate.

2) a) Assume users A and B use the Diffie-Hellman key exchange technique with a common prime $q=11$ and a primitive root $g=2$

i) if user A has the private key $X_A = 8$. What is A's public key Y_A?

ii) If user B has private key $X_B = 4$. What is B's public key Y_B?

Let A chooses a random no $X_A = 8$

iii) What is the shared secret key?

A computes public key Y_A for B

$$Y_A = g^{X_A} \bmod p = 2^8 \bmod 11$$

$$= 256 \bmod 11$$

$$Y_A = 3$$

Let B chooses a random no $X_B = 4$

B computes a public key Y_B for A

$$Y_B = g^{X_B} \bmod p$$

$$= 2^4 \bmod 11$$

$$= 5$$

PAGE NO.

Secret key k by A $K = (Y_B)^{X_A} \bmod p$

$= 5^8 \bmod 11$

$= 4$

Secret key k by B $K = (Y_A)^{X_B} \bmod p$

$= 3^4 \bmod 11$

$= 4$

b) Assume El gamal scheme is used by Sender A and Receives B with a common prime $p=19$ and $g=10$

i) Let A's private key, $a=5$. Compute the corresponding public key α

ii) Let B choose random integers $r=6$,

If B perform encryption for the message $M=17$, Compute C_1 and C_2.

iii) Decrypt the Cipher C_1 and C_2 and obtain the message M

$p=19 \quad q=10 \quad a=5 \quad r=6$

$a=5$

Public key of A $= (p, q, \alpha)$

$\alpha = g^a \bmod p$

$= 10^5 \bmod 19$

$\alpha = 3$

public key of A $\{19, 10, 3\}$

Encryption $C_1 = g^r \bmod p$

$= 10^6 \bmod 19$

$C_1 = 11$

$c_2 = (M \cdot k) \bmod p$

$\left.\begin{aligned} &= (c_1)^9 \bmod p \\ &= 11^9 \bmod 19 \end{aligned}\right\} \times$

$c_2 = (17 \times 7) \bmod 19$

$c_2 = 5$

Decryption by A $\quad k = (c_1)^a \bmod p = (11)^5 \bmod 19 = 7$

$k = 7$

$M = (c_2 k^{-1}) \bmod p$

$= 5 \times 7^{-1} \bmod p$

$= (5 \times 7^{-1}) \bmod 19$

$= (5 \times 7^{-1} \bmod 19) \bmod 19$

$= (5 \times 11) \bmod 19$

$= 55 \bmod 19$

$M = 17$ //

c) List the steps involved in EL Gamal Encryption and Diffie Hellman Exchange Protocol.

- EL Gamal Encryption

EL Gamal encryption uses a large prime number p and a generator g in $(Zp^*, *p)$. An EL Gamal private key is an integer, a, $1 < a < p-1$

The corresponding public key is the triplet (p, g, α) where α is the encryption key calculated from

$$\alpha = g^a \bmod p$$

Let (p, g, α) be the public key of A. To encrypt a message $m < p-1$ to be sent to A, B does the following:

- He chooses a random number r, $1 < r < p-1$ such that r is relatively prime to $p-1$
- He computes

$$c_1 = g^r \bmod p$$

and uses α from A's public key to compute

$$C_2 = (m * \alpha^r) \bmod p$$

- He sends the ciphertext (C_1, C_2) to A.

To decrypt the ciphertext (C_1, C_2), A uses her private key, a and computes

$$(C_1^{-a}) * C_2 \bmod p$$

$$(C_1^{-a}) * C_2 \bmod p = (g^r \bmod p)^{-a} * (m * \alpha^r \bmod p)$$

using the definitions of C_1 and C_2

$$= (g^{-ar} * g^{ar} * m) \bmod p$$

$$= m$$

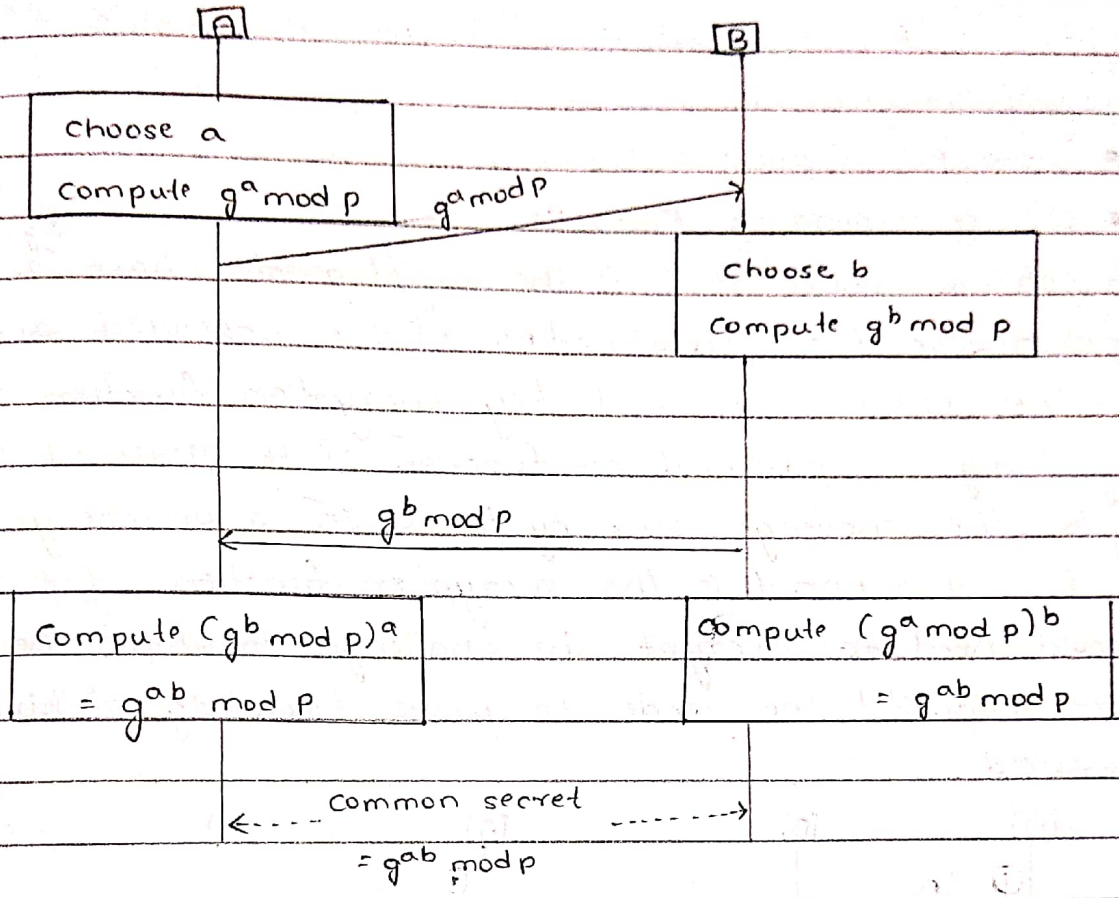
using the definition of the EL Gamal public key.

- Diffie Hellman Exchange Protocol.

Consider two parties, A and B that need to agree upon a shared secret for the duration of their current session. One widely used method is the Diffie-Hellman Key Exchange. For simplicity, assume that both A and B know the base g and modulus p in advance. They then participate in the following sequence

- A chooses a random integer a, $1 < a < p-1$, computes the "partial key", $g^a \bmod p$ and sends this to B
- B chooses a random integer b, $1 < b < p-1$, computes the "partial key", $g^b \bmod p$ and sends this to A
- On receipt of A's message, B computes $(g^a \bmod p)^b \bmod p = g^{ab} \bmod p$
- On receipt of B's message, A computes $(g^b \bmod p)^a \bmod p = g^{ab} \bmod p$

Now, both A and B share a common secret, $g^{ab} \bmod p$.



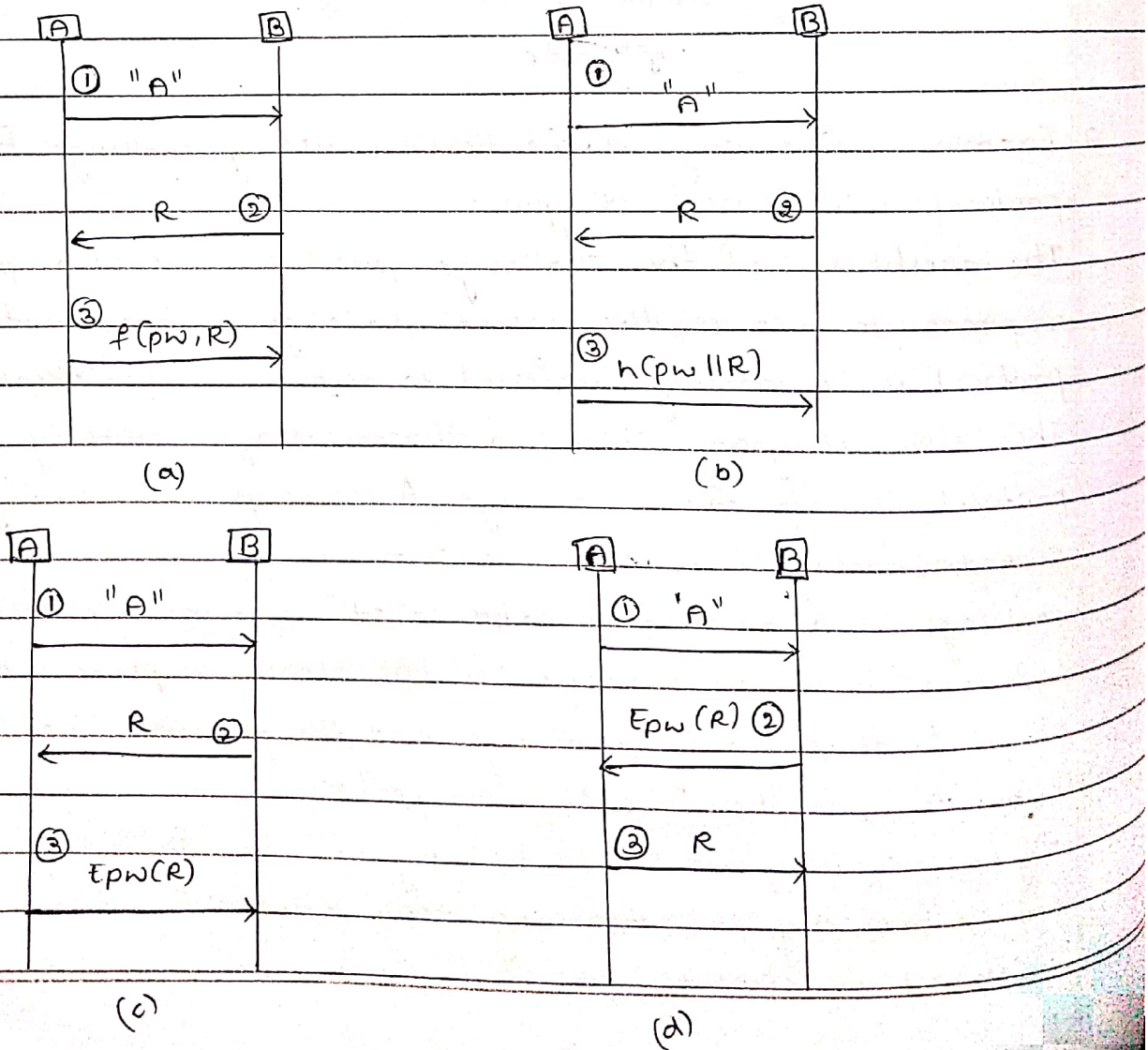
3) Examine One-way Authentication using challenge-Response protocol with a neat diagram.

The freshness of the challenge precludes use of a previous response to answer the current challenge. Such an authentication protocol is commonly refessed to as a Challenge-Response Protocol. The below diagram shows a three-message one-way authentication protocol. In the first message, A conveys its identity. The second message contains the challenge from the server. The challenge is a random numbes called a nonce in security parlance. The third message is the client's response - a cleverly chosen function of the challenge and the password. The function, $f(pw, R)$, has the following properties:

- Given x and y, it should be easy to compute $f(x, y)$
- f is one-way, so, knowing $f(pw, R)$ and R, it should be infeasible to compute pw

- Given an R, it should be infeasible to compute $f(pw, R)$ even if one knows
  - $f(pw, R_1), f(pw, R_2), f(pw, R_3) \ldots$
  - the corresponding R_1, R_2, R_3

An obvious choice for f is the cryptographic hash fig (b) which is applied over the concatenation of the password and the nonce. Another choice is a secret key encryption function with the key being the password or function of the password fig(c). In fig (d), the challenge sent by the server is an encrypted nonce so the function f is the decryption function - the client would need to decrypt the challenge to obtain the nonce and return it to the sender to prove knowledge of his/her password.



4) Evaluate Flawed protocol and Parallel session attack based on Asymmetric key-based Authentication with a neat diagram.

Assume both A and B have public key/private key pairs

In the protocol of fig(a) each party transmits its own nonce and challenges the other to sign it. We use the notation $[m]_A$ to mean a message, m, sent in the clear together with A's signature on m

In Message 2, the string obtained by concatenating nonces R_A and R_B is signed by B. Both the nonces and the signature are sent. Nonce R_A is the challenge provided by A. R_B is the challenge provided by B and signed by A in response (Message 3).

These appears to be a subtle way in which this protocol can be abused as demonstrated by the following attack scenario in fig(b).

Message 1: A initiates communication with C, sending her challenge R_A

Message 1': C initiates communication with B using the same nonce R_A supplied by A

Message 2': B responds to "A's challenge" and includes a challenge of his own, R_B.

Message 2: C responds to A's challenge and uses B's nonce, R_B, as his challenge to A

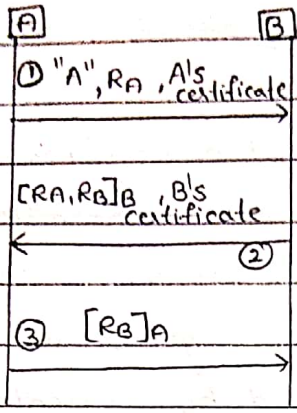
Message 3: A responds to C's challenge. A thus completes the mutual authentication protocol with C

Message 4: C forwards A's response to B

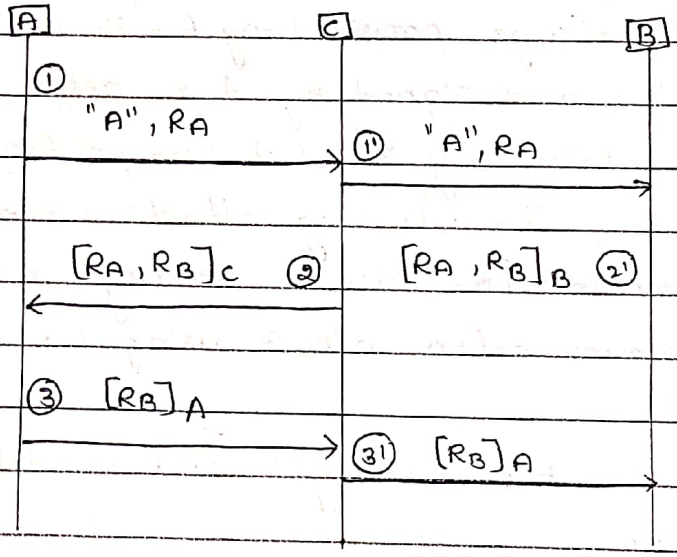
It is clear from fig(b) that

- A does intend to communicate with C
- B wishes to communicate with A. Otherwise, B would not have responded in Message 2' to the nonce presented in Message 1'.

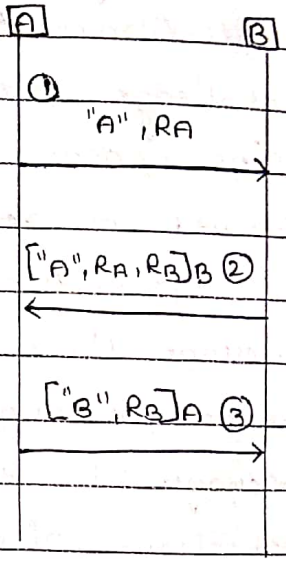
Message 1' is sent by C but it includes A's identity.



Flawed protocol



Attack on flawed protocol



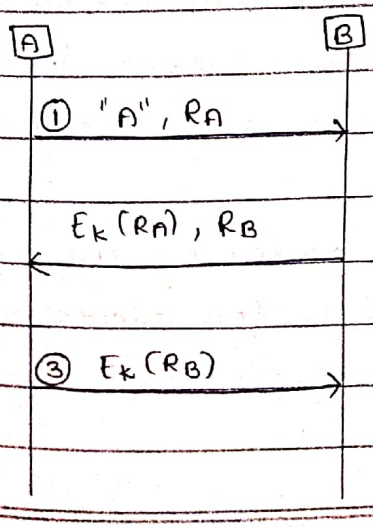
Corrected protocol

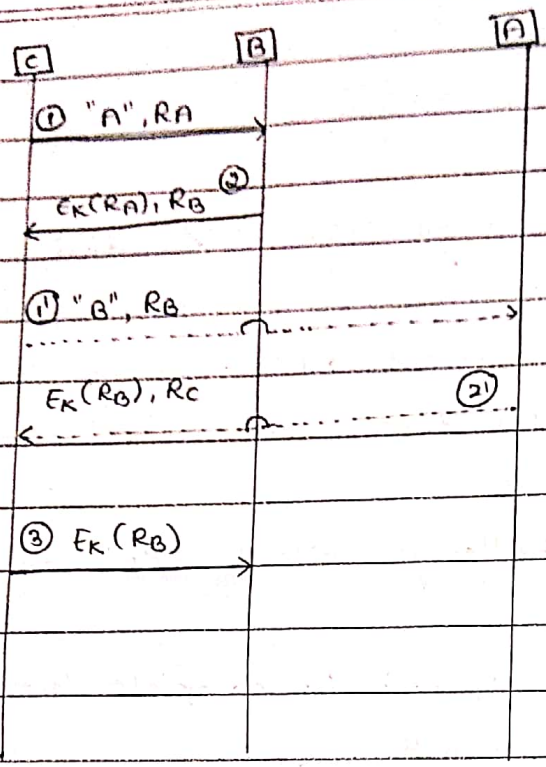
5) Evaluate Flawed protocol and parallel session attack based on Shared secret-based Authentication with a neat diagram. In a Message 1, A communicates its identity and its challenge in the form of a nonce R_A. In message 2, B responds to the challenge by encrypting R_A with the common secret, K, that A and B share. B also sends its own challenge, R_B, to A. A's response to B's challenge in the third message appears to complete the protocol for mutual authentication. While the protocol may appear sound, there are some serious flaws in it.

One attack scenario is as follows.

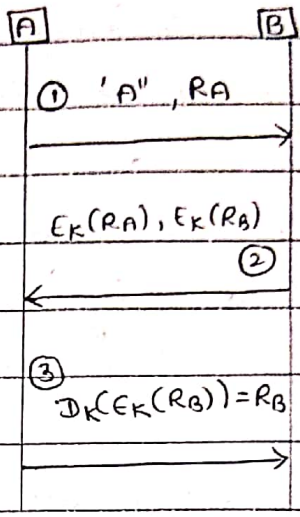
- Message 1: An attacker C, sends a message to B containing a nonce R_A and claiming to be A.
- Message 2: B responds to the challenge with $E_K(R_A)$ and its own challenge R_B as required by the above protocol
- Message 1: Now C attempts to connect to A claiming it is B with the challenge R_B. Note that this is the same challenge offered to it by B in Message 2.
- Message 2: A responds to the challenge with $E_K(R_B)$ and a nonce of its own.
- Message 3: C uses A's response $E_K(R_B)$ to complete the three-message authentication protocol with B

C has successfully impersonated A to B. Message 3 was required to complete the authentication of C to B. However, C could not compute the response to B's challenge since that required a computation involving the secret key K, shared between A and B, C initiated the authentication protocol with A presenting to A the same challenge it had received from B. A's response to the challenge in Message 2' was used by C to convince B that it was A that was trying to establish communication with him.





Parallel session attack



Corrected protocol

6) Inspect dictionary Attack with example:

Dictionary attacks are typically launched in the context of passwords. Some passwords have too few characters

There are two types of dictionary attacks → on-line and off-line

In on-line attacks, an intruder attempts to login to the victim's account by using the victim's login name and a guessed password.

Offline dictionary attack leaves leaves fingerprints. One possibility is for the attacker to get a hold of the password file. Another possibility is for the attacker to eavesdrop on the communication link during client authentication.

Eg: consider // Let D be an array containing the dictionary
// Let F denote f(pw, R) where pw is the client's password
// Let n be the no. of permissible guesses

found = false

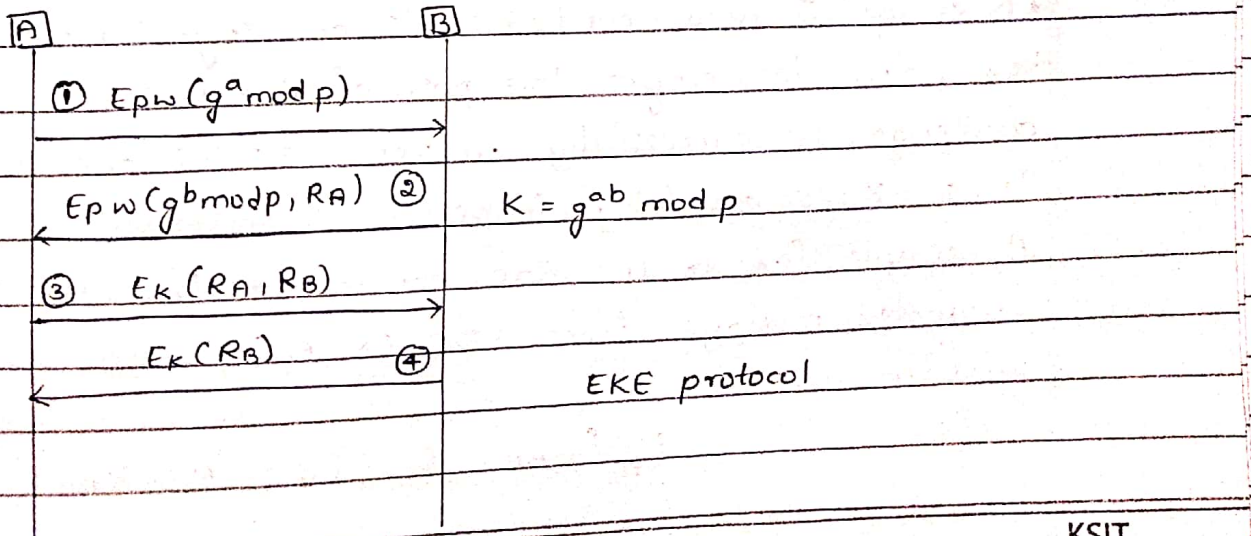
i = 0

```
while (!found && i<n)
  {
    x = f(D[i], R)
    if (x == E)
      { print(" CORRECT PASSWORD is D[i]")
        found = true
      }
  }
```

The most time consuming operation in each iteration of the dictionary attack program is f (D[i],R). Hence to decrease the Attacker's chance of success, the function f(D[i], R) could be made computationally expensive. h (.... h (h (D[i], R)) ...)

A protocol that virtually eliminates off-line dictionary attack is Encrypted Key Exchange (EKE) protocol. This is a password-based protocol that combines Diffie-Hellman Key exchange with mutual authentication based on a shared secret.

The below figure shows the four messages that are exchanged in EKE. After Message 2, both sides should be able to compute the new session key = $g^{ab} \bmod p$ denoted by K. EKE is not susceptible to an off-line dictionary attack. Another property of EKE is that it provides perfect forward secrecy.



EKE protocol

7) Inference Needham - Schroeder protocol - Preliminary version 1 with Man-in-the middle attack to provide mutual authentication with a neat diagram

The below figure enhances the protocol i.e message confidentiality using KDC to provide mutual authentication by including a challenge - response phase (Message 3, 4 and 5). Here, both sides proceed to challenge the other to prove knowledge of the session key, K_{AB}. The challenge is a nonce. The response involves decrementing the nonce and encrypting the nonce with the session key, K_{AB}

Preliminary Version 1

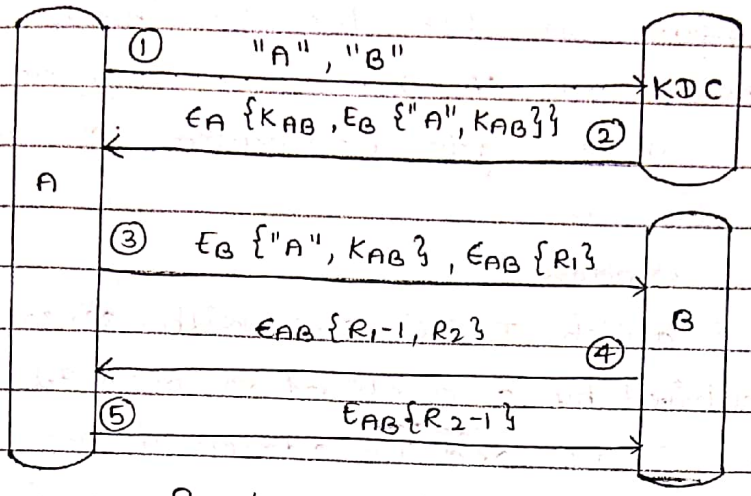
The protocol in fig is susceptible to an impersonation attack shown in fig(b). The attacker, X, is an insider who shares a long-term key with the KDC

- The attacker, X, intercepts Message 1, substitutes "B" for 'x' and sends the modified message to KDC (Key Distribution Centre)
- In response, the KDC creates a ticket encrypted with x's long term key and sends it to A in Message 2
- Now X intercepts Message 3. He decrypts the ticket using the long-term secret he shares with the KDC. He thus obtains the session key, K_{AX}
- Message 3 also contains A's challenge R_1. X uses the session key, K_{AX} to decrypt the part of the message containing A's challenge. He successfully responds to A's challenge in Message 4

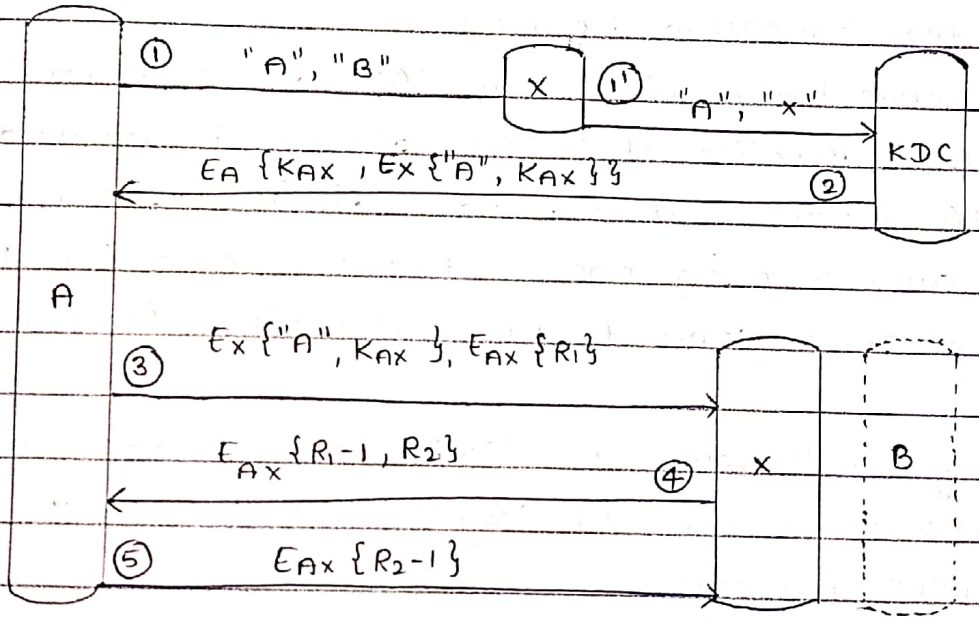
Thus, X successfully impersonates B to A

A simple fix to the protocol include B's identity in the encrypted message from KDC to A (Message 2). The modified message is

$$E_A\{K_{AB}, \text{"B"}, E_B\{\text{"A"}, K_{AB}\}\}$$



Preliminary Version 1



Man-in-the middle attack on preliminary version 1

8) List the fields of IPSec Security Association.

- Lifetime of the association
- IPSec Mode - transport or tunnel
- Cryptographic parameters (algorithm used for encryption, if any, and for computing the integrity check, together with the keys)

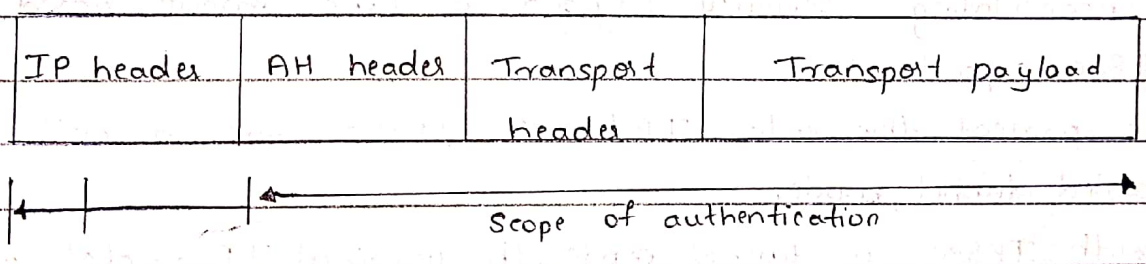
- A 32-bit sequence number - the first packet produced by a newly established SA bears the sequence number 1, the sequence number gets incremented for each new packet sent.
- An anti-replay window
- A node may establish IPSec SAs with several nodes. An SA is uniquely identified by a combination of a 32-bit Security Parameter Index (SPI) and IP address of the connection endpoint. Each IPSec packet contains a value of SPI in its header. This is used by the receiving node to identify SA to be used for processing the packet.
- Each node has a database of SAs for all connections originating from or terminating at it. This database is referred to as the SA Database (SADB). Finally it should be noted that two communicating parties, A and B, establish two SAs - one for communications from A to B and another from B to A

9) Distinguish between AH (Authentication Header) and ESP (Encapsulating Security Payload) in transport mode based on IPSEC protocol.

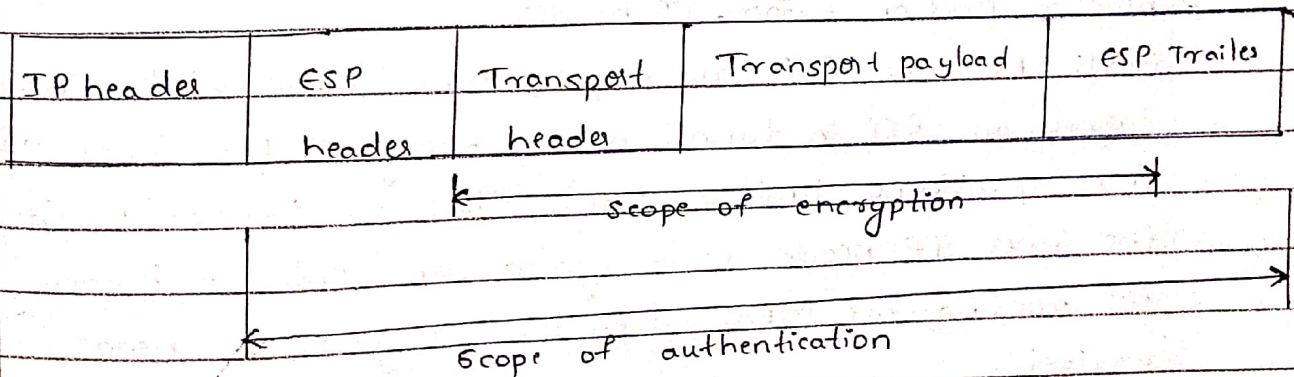
- The below figure shows the headers introduced by AH and ESP i.e Authentication Header and Encapsulating Security Pay-load. The IPSec header is sandwiched between the IP and TCP headers. Message authentication and integrity are provided by the use of a keyed MAC in both cases. While the MAC is a part of AH header, it is included in the trailer of an ESP packet.
- IPSec implementations are required to support MACs based on MD-5 and SHA-1. However, the MAC field in the IPSec packet is only 96 bit. So, only 96 most significant bits of the computed MAC are actually transmitted.

- With AH, the integrity check is computed on parts of the IP headers such as the source and destination IP addresses. Mutable parts of the headers such as TTL, TOS and header checksum fields are zeroed before computing the HMAC. By contrast, ESP does not provide protection to any part of the IP headers in transport mode.
- All IPSec headers have 32-bit fields each for the SPI and a packet sequence numbers. The latter was intended to protect against replay attacks. Padding is added so that the length of encrypted payload is a multiple of block size. Padding also helps in hiding the actual length of the data when ESP is used with the encryption option turned on.

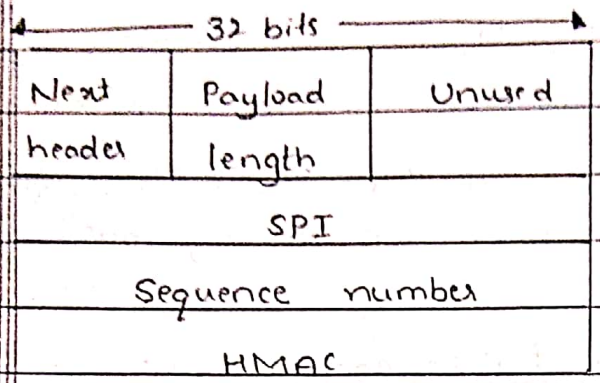
AH



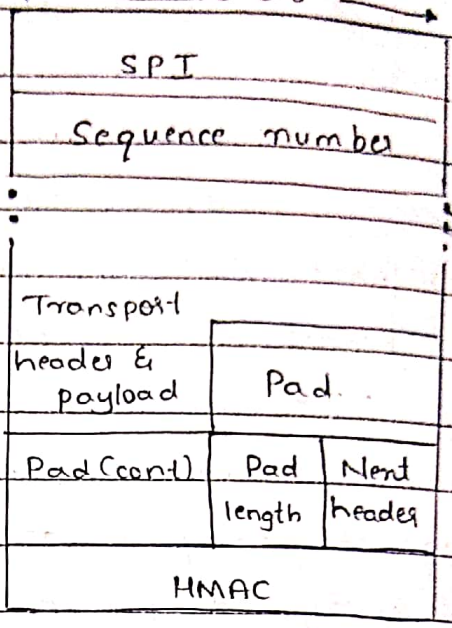
ESP



AH header



ESP header



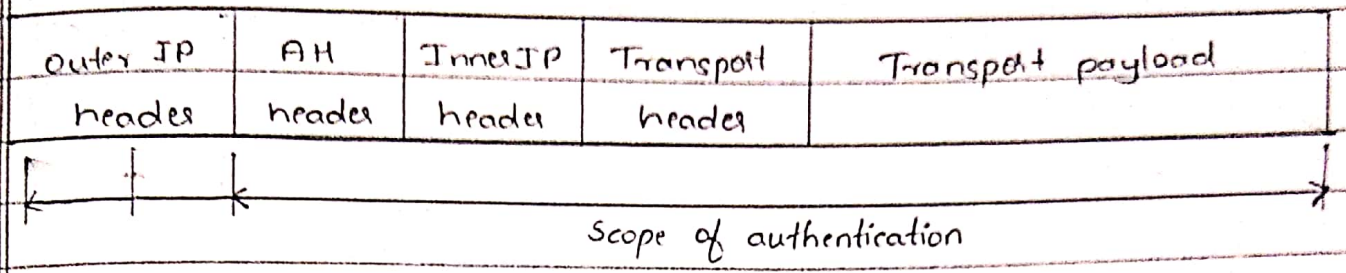
AH and ESP in transport mode. ESP Trailer.

10> Distinguish between AH (Authentication Headers) and ESP (Encapsulating Security Payload) in tunel mode based on IPSec protocol.

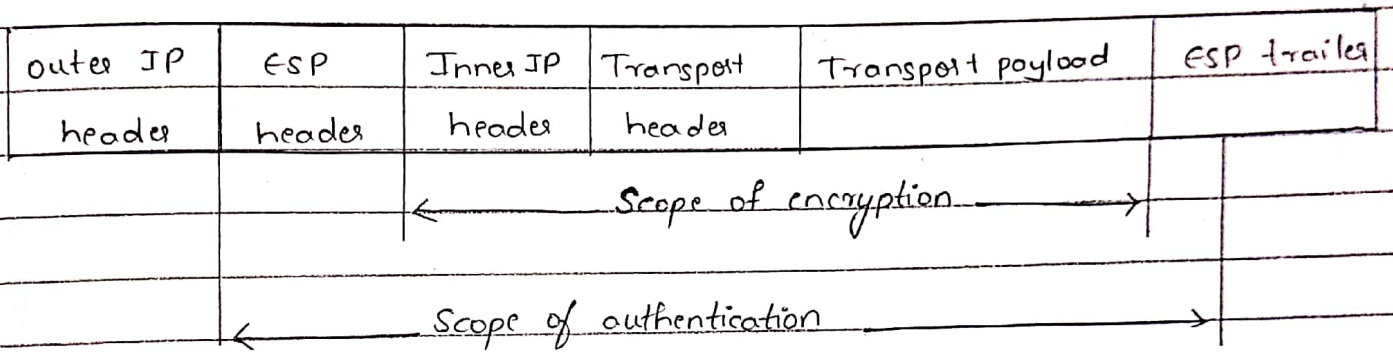
- To protect the entire IP header, IPSec has an option called tunnel mode.
- With IPSec in tunnel mode, the original IP packet is encapsulated within larger packet containing an IPSec header and an extra IP header.

Both AH and ESP can employ tunnel mode. With encryption turned on, ESP in tunnel mode encrypts the "inner" IP header thus providing limited traffic flow confidentiality. Because the inner IP header is encrypted, an "outer IP header" is used for routing. The MAC covers the original IP header in its entirety.

AH



ESP



AH and ESP in tunnel mode.