"QT"

Date
Page

☆ Key Management:

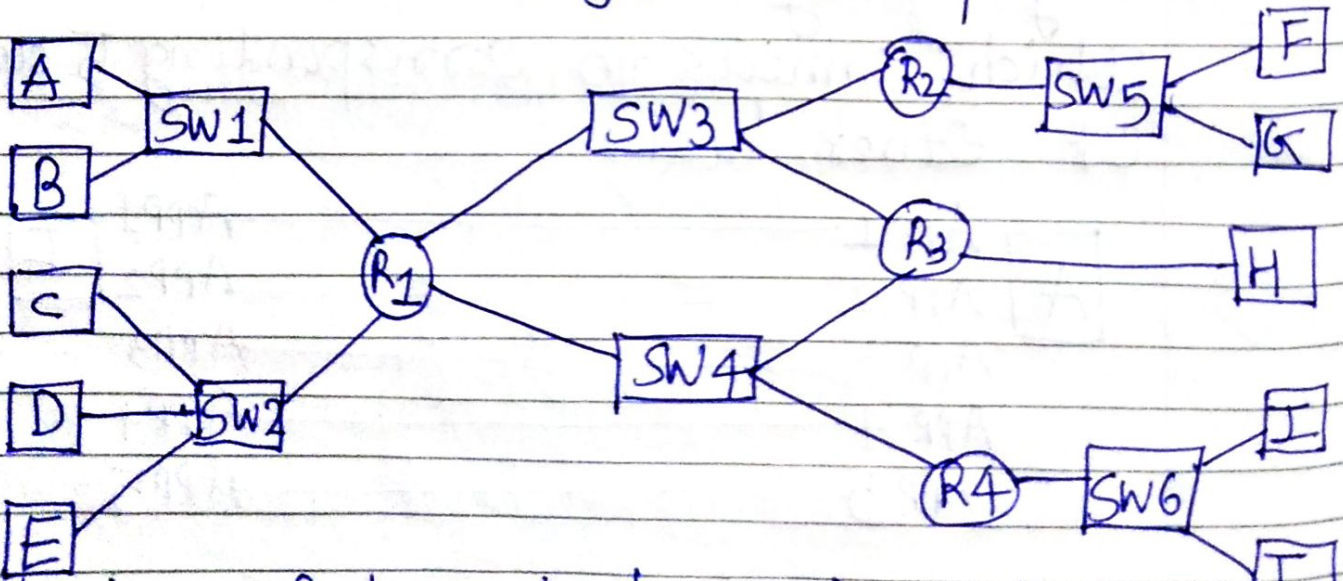
① How to share a secret key?
→ We know DH key Exchange

② How to Obtain someone else's public key?
③ When to Change keys?

☆ Assumption and Principles

→ Many user's wish to communicate securely across network.
→ Attacker can intercept any location in Network.
→ Manual Interactions between users are undesirable.
→ More times a key is used, greater chance for attacker to discover key.

☆ Where should Encryption be performed?



→ Number of keys to be exchanged depends on number of entities wishing to communicate

→ Related Issues:

Where to perform Encryption?

Way 1
Encrypt separately across each link

Way 2
Encrypt only at end points

→ Let's assume that We use Secret keys (Symmetric Cipher key) to encrypt and decrypt the data.

→ Way 1: If Way 1 is used then there is key for every link
→ At one end point of the link encryption happens with the secret key (of link)
→ At the other end decryption will happen.
→ This other end becomes the first end for the next link.
→ This will encrypt using new key which is shared between this node and the node at the next end point.
→ Similarly for all

How many Secret keys are needed to be shared?

Answer: As the Number of Links = 20.

☆ What are disadvantages of Way 1?

→ Message is encrypted/decrypted once for each link.
∴ Computational cost and time is more.

→ At each link, message is open.
∴ If Link is not reliable, one can get the messages at that end points (Switch/Router etc.)

→ We need all the devices capable of Computing Encryption, Decryption function with reliability.

↓ Then why should we go for way 1?

Reason will be clear after Way 2.

Way 2: End to End Encryption:

Entities: Host: A, B, C, D, E, F, G, H, I, J

→ We have Encryption/Decryption only at ends

∴ Encryption at say A and Decryption at say F (If A: Sender, F: Receiver)

∴ We need

$$9+8+7+6+5+4+3+2+1$$

$$= \frac{9*10}{2}$$

$$= 45 \text{ keys}$$

Note: Both ways same key is assumed i.e. A → F, F → A use same key.

Compare: End to End : 45 keys
Link Level : 20 keys

★ Note: Generally we have applications running at every host which communicates with applications running at other host.

→ Say every host has 5 applications which connects to corresponding 5 applications of other host.

Now Number of entities (Applications) which need to communicate with other entities will increase in End to End.

∴ For end to end, Number of keys required =

Number of pairs of hosts * # of apps / host (which want communication)

= 45 * 5

= 225 keys

If I allow any application wants to communicate with any other application, then

for end to end

# of keys = ~~Pairs of hosts~~ 45 ~~* 25~~ $\frac{50 * 49}{2}$

= 1225 keys

why?

1 host * 5 applications / host

= 50 applications

= 50 entities

∴ # of keys = $\binom{50}{2} = \frac{50 \times 49}{2} = 1225$ keys

∴ We can see that even for smaller network like this we require 1225 keys.

→ If Number of entities = 1000 then

$$\#\text{of keys} = \frac{1000 \times 999}{2}$$

$$= 500 \times 999$$

$$= 499500$$

$$\sim \underline{5,00,000} \text{ keys}$$

∴ End-to-End encryption has drawback of large number of keys.

→ These keys need to be shared and administrators can't do this manually.

↓ why?

Keys are not Permanent.

↓

They need to be changed periodically

↓

∴ Manual way is tedious and not practical

∴ We need efficient way for key Distribution.