Key Management & Distribution

→ We are concerned about sharing keys between sender & receiver in cryptography.

→ To provide secure key distribution

→ There are situations we need to deploy two keys –
Master keys – not frequently used but are longlasting
Session key – Temporary keys. not longlasting.

→ Provide certificates are used to provide authentication & other security services.

→ To do this we need PKI.
To manage people, policies H/w S/w procedures.

→

Key Mangement & Distribution.

1) Symmetric KD. using symm. encryption.
2) " " using Asym. "

When we talk about Key Distribution it is about ~~key~~ symm. encryption but for acheiving this we have these two options.

Various ways of Distribution of Public keys -

- Public announcement
- Pulically available directory
- Public-key authority
- " " Certificates

X.509 Certificates

It based on asymmetric (Publickey) Cryptography. It defines a framework for the provision of authentication service. Not specific algorithm for encryption & hashing.

Symmetric Key Distribution using Symmetric encryption

→ Sharing of same key.
→ Keys must be protected from access by others - only sender & receiver can access.
→ Keys must be changed frequently and communicated to other.
→ When remote users are communicating the key distribution technique is important.

Various ways of Key Distribution

A. Physically delivery it to receiver
B. Third can decide the key & physically deliver it to A & B.

C. If physically delivery is not possible
→ use a new key, sending this new key encrypted using previously used old key. (which was earlier used by communicating parties)

D. Trusted third party can deliver the key. to A & B. on encrypted links.

Drawbacks

In distributed system, one node can connected with 10 different other nodes, this node require 10 different key for secure communication.

Like wide area network, large no. of parties are communicating.

No. of required keys = $N \times (N-1)/2$.

Suppose 2 parties are communicating

$$2 \times (2-1)/2 = 1$$

1 key required.

In reality it is a challenge because many nodes are communicating.

$(1000 \times 999)/2 = 999000/2 = \underline{499500}$

The use of key hierarchy

Data is encrypted by session-keys. Session keys are valid for particular session. and are temporary. Once these keys are used then these keys are discarded.

These session keys are provided by key distribution centre.

These session key are required to be distributed among sender & receiver, for secure transmission these keys are encrypted by master keys.

A Key Distribution Scenario

Step 1. Initiator A will contact KDC to get the key. & provide the identifier information —

$$\underbrace{ID_A}_{\text{own identity}} \,||\, \underbrace{ID_B}_{\text{receiver's identity}} \,||\, \underbrace{N_1}_{\text{Random number}}$$

N_1 is required to prevent Masquading attack. N_1 is different for each request.

A pretends other person.

N_1 is sent back from receiver, like Digi

Step 2 - KDC is replying as -

$$E(K_a, [K_s \| ID_A \| ID_B \| N_1 \| E(K_b, [K_s \| ID_A])])$$

Part 2

K_a → key → KDC & A knows.

K_s → Session key.

$ID_A \| ID_B \| N_1$ → (received from A)

Part 1

K_b → Shared secret key. Shared between B & KDC not A

$[K_s \| ID_A]$ → ~~Letting receiver~~ This A will send to B.

In step 2. KDC will make a reply for A. In which there are 2 parts, ~~and~~ ~~both~~ Both the parts are encrypt-ted using K_a, which is a secret key shared between A & KDC. When A will receive A can decrypt it. First part K_s is session key, which is requested by A. and with this K_s, other is the info received ~~to~~ from A.

In part 2, the session key K_s & ID_A of is provided for receiver which is encrypted through K_b, K_b is shared secret key b/w KDC & B, (which is not accessible to A)

Step 3

A will send message to B. received from KDC, once B. receive it B can decrypt it.

Step 4

They have to agree on key so B will confirm K_s and send it with N_2 random number. To protect it from Masquerading.

Step 5

A will send same session key & function of N_2. for example $N_2 = 100$ earlier, now A will send 101 as N_2. using (+1) as function.

In whole scenario the session keys are distributed, & sender & receiver are mutually agreeing on same keys.

Automatic Key distribution

Whenever any application wants to send some data to other host in secured way.
In step 1, Application request security service to get the connection with KDC.
In step 2, security buffers the packets received from app. & also same time contacting KDC for session key.

In step 3, KDC provides session key to both the hosts, (receiver & sender) because in connection oriented, connection is established first before communication starts.

Step 4 : sender & receiver can communicate using session key.

Session keys are valid for particular session only.

Decentralized key control

For the situations where KDC is not there in scenario, there is no Central entity to provide keys. However, this ~~case~~ is not possible in case of large networks, which is using Symmetric encryption.

We can achieve this by master keys. In step 1, ID_A & N_1 random number are provided by sender to receiver directly

In step 2, receiver B, responds, in which N_1 is transformed by using some function $f(N_1)$ and also send N_2 (Given random no.) Identifier information of A & B. Along with this B receiver send K_s (Session key) and whole information is encrypted using Master key K_m, which is known to A & B. Master keys are permanent keys.

Basically in this step, B is sharing session key with A.

In step 3, A is sending $f(N_2)$, which is transformed N_2 and same key K_s to mutual agreement.

Symmetric Key Distribution with Asymmetric Encryption.

A will send his public key with A's identifier.
B will reply with K_s (session key) encrypted with public key of A.

After decryption A will have shared session key K_s.

This technique is vulnerable to Man in the middle attack.

Alice (sender) is telling ID_A & PU_A, Attacker will receive this information
Darth (attacker) will generate his public & private keys. and send receiver ~~to~~ Darth's public key with ID_A. Bob (receiver) will share public key of Darth & session key. which is encrypted using PU_{Darth}.
Now Darth send K_s to Alice with ~~do~~ ~~Darth's~~ ~~pub~~ Alice public key.

The Confidentiality and authentication are lost in this process.

To stop this N_1 & N_2 Nonce are used and even the identifier of A is encrypted using receiver's public key which can only be decrypted using B's private key.

Various Public Key Distribution Techniques

1. Public Announcement

Every participants announce the their public key by any means.

2. Public available directory

Who want to publish its public key will request to public key directory to register itself and when anyone wants to know the public key of other participant will access the public key directory

If A want to change the public key later can update and replace its keys anytime.

Public key directories are more secure ways to share public key as compare to public key announcements. Also changing public key is easy in this technique as compare to previous technique.

3. <u>Public Key Authority</u>

Public key authority is a central entity which distributes the keys, which maintains dynamic public key directory as in the previous case. (dynamic because keys can be easily replaced by participants anytime)

Each participant should know the public key of publickey authority. This public key authority has its private & public key pair. Whenever anyone contact to authority they should be aware of its public key.

Let see in example A want to send confidential message to B. A needs to get the public key of B.

Step 1. - 'A' sends a timestamp request to authority, to get the B's public key. ~~with the~~ Same timestamps are attached ~~to ensure~~ with the reply also to ensure message is authentic.

In Step 2. authority responds with public key of B and same request & Time stamp and the response is encrypted with ~~public key~~ private key of authority because A can decrypt it with its public key. (authority's).

Step 3 A now can send message to B to ensure authenticity N1 (Nonce) a random no. is added with message ~~to~~ ~~this~~ N1 will be sent back to A, to confirm that message is from B only.

When B receives the message from A B also contact the authority in the same way to get ~~the~~ A's public key. B also append N_2 to ensure authenticity.

4. Public Key Certificates

In this technique public key authority is replaced with Certificate authority. Instead of giving public keys it issues certificates.
Everyone shares its public key with authority. And in response authority will provide certificates. and when A want to communicate with B it directly send certificate with B. Both shares the certificate to ensure ~~auth~~ public keys.
When certificate authority share the certificate it ~~send~~ is encrypted using private key of certificate authority.

Certificate authorities are trusted third party.

X.509 is example of certificate authority

You can check Google's certificate by clicking the ~~lock~~ symbol just before the URL Google.com, by clicking on "connection is secure"

Kerberos

When any user wants to access his workstation's files from remote location, workstation will give access to files to only authentic user because any attacker can also try to access.

It is a authentication protocol. (Whenever sender & receiver want to communicate, they need to authenticate themselves)
Kerberos works based on tickets

Firstly, the user request for ticket to A.S., AS will check Database (ID/PSWD) and reply with Ticket + session key.
When user got the ticket, he/she request for service granting ticket to TGS.
User can request ~~no~~ other requests for services, he need not to request for ticket every time, for ticket, a user have to ask once for a session.
Once the user will get the ticket from TGS, user can request for a service

Server reply with service as well as server authenticator to ensure that server is authenticate.