Key Distribution Problems

Public key cryptography depends on the distribution of keys. Everyone having to store everyone else's keys scales badly. One way to solve this problem is by having centralised databases of keys. These are called *Key Distribution Centers* or (KDC).

Key Distribution Centers

Key Distribution Centers have the following properties

- Based on Secret-Key Systems
- Every user shares K_A with the KDC
- When A wants to talk to B, KDC generates $E_{K_A}(K)$ for A, $E_{K_B}(K)$ for B.
- All users must trust KDC completely
- If the KDC is down or compromised, the entire network is fåkked.

Certification Authorities

We can replace the previous example with the following protocol

- Key K_{AB} is replaced by pair (sk_B, pk_B). Here pk_B is public.
- A sends session key $E_{pk_B}(K)$ to B.

How do we distribute these? Assume we have a *Certification Authority* CA with the pair (sk_{CA}, pk_{CA}). Everyone then gets a copy of pk_{CA}. Then begins the hard part of authenticating yourself with the CA. This might even be done in person. The CA then issues cert ID_A, and pk_A and then the signature $S_{sk_{CA}}(ID_A, pk_A)$. Now everyone can check to see if the public key is legitimate.

Certificate Chains: Or Who Verifies The Verifiers

If two people do not have certificates issued from the same CA, they can use what's called Certificate Chains to solve this problem. If CA's certifiy each other's public keys. If user A has CA_1 and B has CA_2. If A receives $Cert_{CA_2}(B, pk)$ then if he also gets $Cert_{CA_1}(CA_2, pk_{CA_2})$ then A can verify that certificate. A chain has the from

$$Cert_{CA_1}(B, pk_B), Cert_{CA_2}(CA_1, pk_{CA_1}).....$$

It should be noted that this requires one to trust the initial part of the chain in the first place.

What they know: Passwords

Passwords identity users from what they know.