Review

- RSA

\[
\text{Alice} \quad S_{\text{Alice}} \quad P_{\text{Alice}} \quad E(P_{\text{Alice}}, M) \quad \text{Bob}
\]

Is this secure?
Eve can impersonate Alice

Man-in-the-middle attack

\[
\begin{align*}
\text{Alice} \quad S_{\text{Alice}} & \quad P_{\text{Alice}} \quad E(P_{\text{Alice}}, M) \quad \text{Eve} \quad S_{\text{Eve}} \quad E(P_{\text{Eve}}, M) \quad \text{Bob}
\end{align*}
\]

- In the above situation, Bob and Alice don’t even know about the attack.

- Bob believes that he is communicating with Alice. Hence, we need to authenticate the sender. Hence, there is a need to bind the public keys to identities or in the other words to authenticate public keys.

- Public key infrastructure:

\[
\begin{align*}
\text{Alice} \quad S_{\text{Alice}} \quad P_{\text{Alice}} \quad (\text{Alice}, P_{\text{Alice}}, \text{Cert}_{\text{Alice}}) \quad \text{Bob} \\
\text{Trent} \quad \text{Cert}_{\text{Alice}} = \text{Sig}(S_{\text{Trent}}, (\text{Alice}, P_{\text{Alice}})) \quad \text{Secure (somehow)} \quad \text{Trusted Third Party}
\end{align*}
\]

In the above situation, Trent somehow will verify the identity of Alice.

- Trent’s signature means that he has verified the identity of Alice and its public key is \(P_{\text{Alice}}\).
• Biggest criticism of this public key system is that Trent is all powerful. If he is malicious then the whole system breaks down.

✓ How to solve the problem of mistaken certificate?
  ○ Certificate revocation list.

Normally there are multiple 3rd parties and if there is a certificate from one of the parties then that is good enough.

✓ PGP web of trust.

Bob will check how much he believes $P_{Alice}$ based on how much he trust Dave and Carol to sign for public keys. If the belief is over a threshold then he gives a certificate to Alice.
Public key signature ($P_{Alice}$ is public information)

$$Y = \text{Sig}(S_{Alice}, M)$$

Ver($P_{Alice}$, $M$, $Y$) = valid

**Security for signature**

A signature scheme consists of sig & ver algorithms such that ver($P_x$, $M$, Sig($S_x$, $M$)) = valid and without knowing $S_x$ it is difficult to compute any pair $M$, $Y$ s.t ver ($P_x$, $M$, $Y$) = valid.

**RSA Signature**

Pick p, q

$$N = pq$$
$$Ed = 1 \mod \phi(n)$$
$$P_{Alice} = N, e$$
$$S_{Alice} = N, d$$

$$S = \text{Sig}(S_{Alice}, M) = M^d \mod N$$

i.e. ver($P_{Alice}$, $M$, $S$) = valid iff $S^e = M \mod N$

This scheme will not work, consider the following case

Given, $M_1$, $S_1 = M_1^d \mod N$
$$M_2, S_2 = M_2^d \mod N$$
then $S_1 \times S_2 \mod N = (M_1 M_2)^d \mod N$

Hence to solve this issue, we hash the message before doing the exponentiation

To sign $M$, compute $h = H(M)$
$$S = \text{Sig}(S_{Alice}, M) = h^d \mod N$$
ver($P_{Alice}$, $M$, $S$) = valid iff $S^e = H(M) \mod N$

As $H(M_1).H(M_2) \neq H(M_1 M_2)$ this scheme works.
**Diffie-Hellman Key Exchange**

Alice – g, p(large prime number)
Bob – g, p

p is prime, (p-1)/2 is prime and \( g^{(p-1)/2} = 1 \mod p \)

Alice picks a random number ‘a’ and Bob picks a random number ‘b’

<table>
<thead>
<tr>
<th>Alice</th>
<th>( Y_a = g^a \mod p )</th>
<th>Bob</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Bob</td>
</tr>
<tr>
<td></td>
<td>( Y_b = g^b \mod p )</td>
<td>K = ( g^{ab} \mod p )</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Encrypt using K</td>
</tr>
</tbody>
</table>

Problem: Given p, g, \( g^a \) computing ‘a’ is the computational Diffie-Hellman problem or discrete log problem and it roughly takes \( O(p^{1/2}) \) time.