Station to Station protocol

→ Original name

- Put together PFS and direct authentication (proving the peer the knowledge of session key, key confirmation).
  - Protects session key from future compromission of shared secrets or secrets of peers

NOTE: $\langle X \rangle_P$: digital signature made by P on X.

- Uses PKE

We assume every peer has a pair of keys.

A ($pubK_A$, $privK_A$) — B ($pubK_B$, $privK_B$)

a ← gen()

M1: g^a →

Alice generates a pair param

b ← gen()

upon receiving g^a, Bob generates b and computes $K_{AB} = (g^b)^a \bmod p$

DELETE b

← M2: g^b, $\{\langle g^b, g^a \rangle_B\}_{K_{AB}}$

This is to authenticate Bob's public param, to avoid MITM. Presence of g^A is a sort of nonce for A, because it is freshly generated a is fresh ⇒ g^a fresh. So Alice realizes message comes from B (signature) and belongs to the current execution.

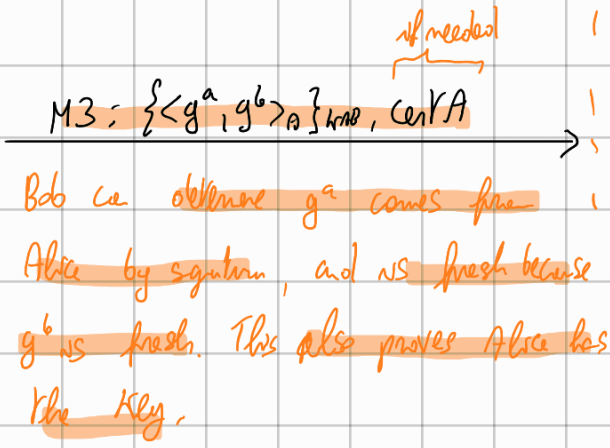
Alice can compute session key: $K_{AB} = (g^b)^a$.

Decrypts CT and verifies the digital signature.

Note: Since $\langle \rangle_B$ proves also g^b comes from Bob, g^a proves freshness, then, the encryption using K_{AB} is there why? $\langle X \rangle_B = X, sign_{privK_B}(h(x))$. After decrypting

for Alice, she checks wether the Cleartext contains qualities she knows. Alice concludes that message has been correctly computed by Bob and he knows session key (key confirmation). Message 2 can also be enriched with certB.

- Now Alice can delete a to avoid compromission.



Bob can determine g^a comes from Alice by signature, and is fresh because g^b is fresh. This also proves Alice has the key.

From now on K_{AB} can be used as session key, and at the end of session it has to be deleted.

- Protocol should not allow an offline Pwd attack.

$S \to C$: n_S ↗ Password. 2nd version of 1st possibility

$C \to S$: $\{n_S, K, n_C, P\}_{TTS}$ ↗ eavesdropping protection

↳ client generates key. Must be an authority.

$S \to C$: $\{n_S, n_C\}_K$

If someone intercepts session, he/she knows n_S. K is random, but attacker knows pubkey. So might try. But K randomizes message: exhaustive PW attack is not possible because for each PW I try I should check all keys.

We could remove n_C because key is fresh.

Adversary wants to discover PW. So C should take CT made of 3 fields:

n_S	K	P

if bro wants to see if P=pippo, bro should encrypt with pubkey K see if encrypted value is correct, should try all possible keys. 2^{128} times.

That's why you can't do pw attack.

1. If protocol allows replay
2. If protocol exposes you to other attacks like offline pw attack.

NOTE: If you suppose K is compromised, you discover K and discover n_S, n_C by M3 and you could make an exhaustive pw check after you discover n_C. Solution: remove n_C from M3, that will act as salt in M2.