Perfect Forward Secrecy

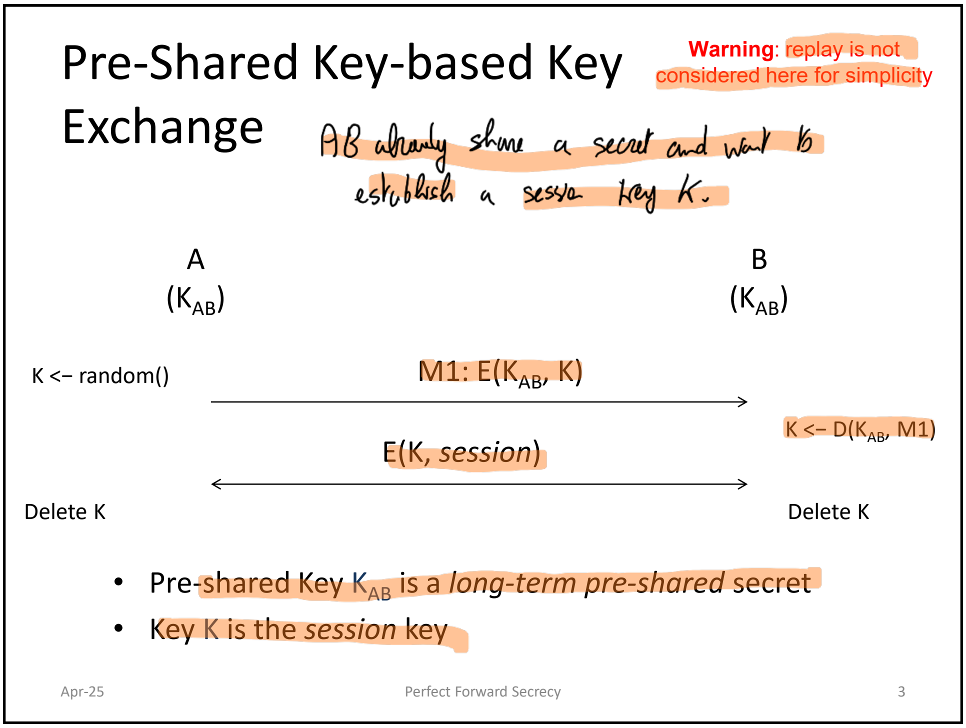
Gianluca Dini
Department of Ingegneria dell’Informazione
University of Pisa
Email: gianluca.dini@unipi.it

Version: 16/04/2025

He who controls the past controls the future. He who controls the present controls the past.

George Orwell

quotefancy



Pre-Shared Key-based Key Exchange

Warning: replay is not considered here for simplicity

AB already share a secret and want to establish a session key K.

A (K_{AB})

B (K_{AB})

K <– random()

M1: E(K_{AB}, K)

K <– D(K_{AB}, M1)

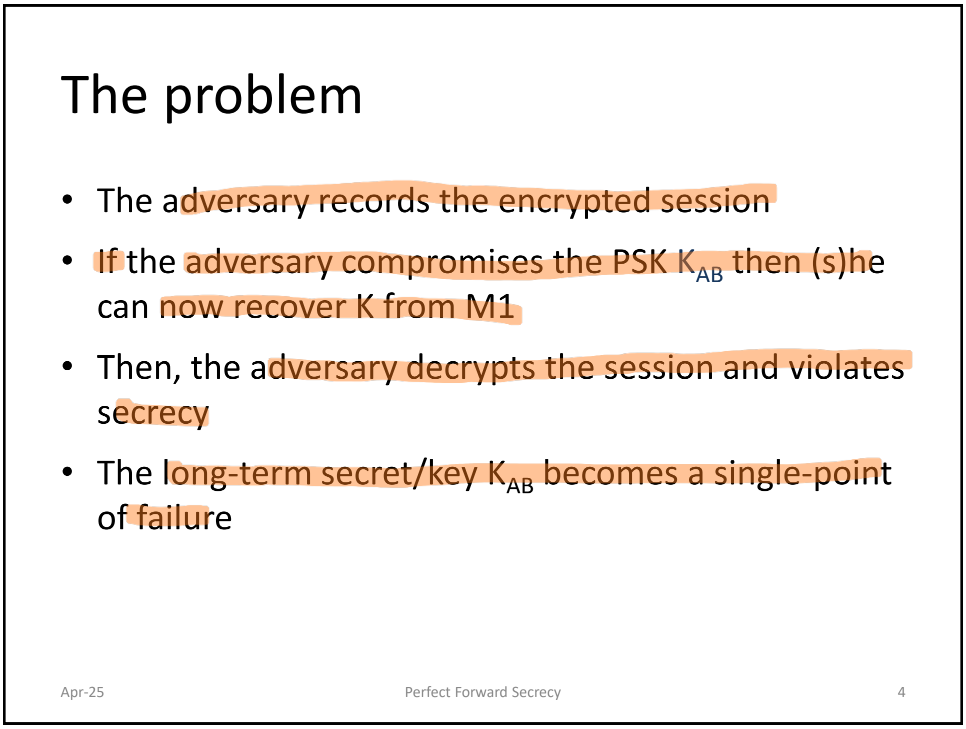
E(K, *session*)

Delete K

Delete K

- Pre-shared Key K_{AB} is a *long-term pre-shared* secret
- Key K is the *session* key

3



The problem

- The adversary records the encrypted session
- If the adversary compromises the PSK K_{AB} then (s)he can now recover K from M1
- Then, the adversary decrypts the session and violates secrecy
- The long-term secret/key K_{AB} becomes a single-point of failure

4

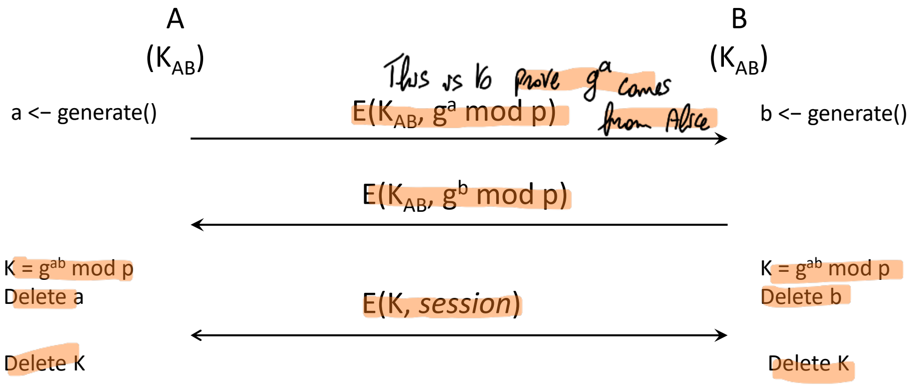
Perfect Forward Secrecy

- **(DEF) Perfect Forward Secrecy**
  - Disclosure of long-term secret keying material does not compromise the secrecy of the exchanged keys from earlier runs
- Public Key Cryptography makes it possible to achieve this requirement

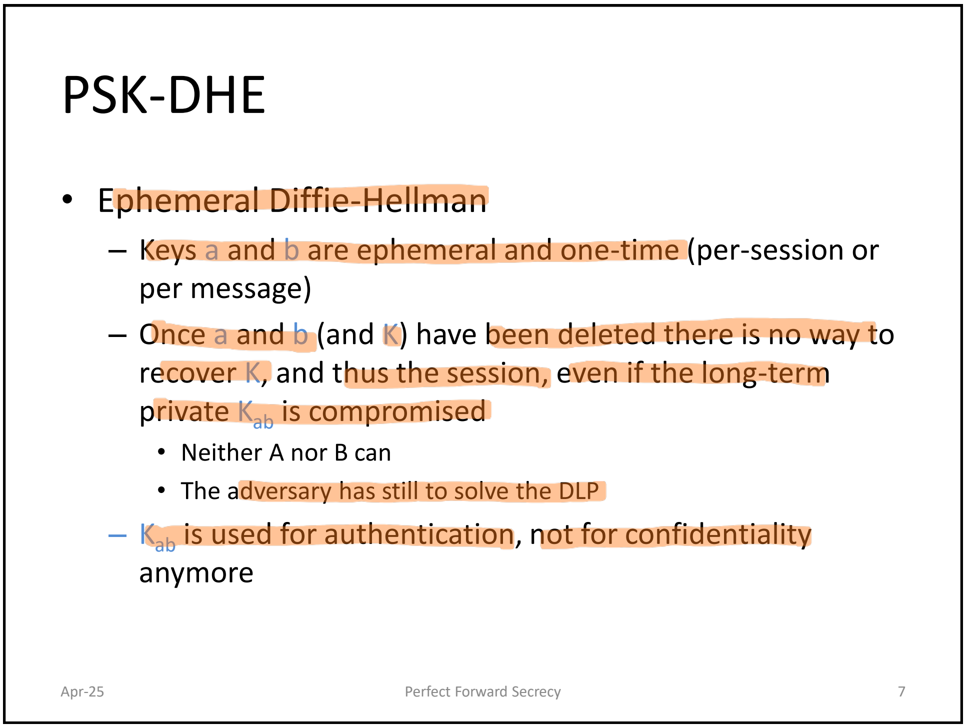
We want to keep the past safe even after discovery

Pre-shared Key – Ephemeral Diffie-Hellman (PSK-DKE)

(p, g) publicly known

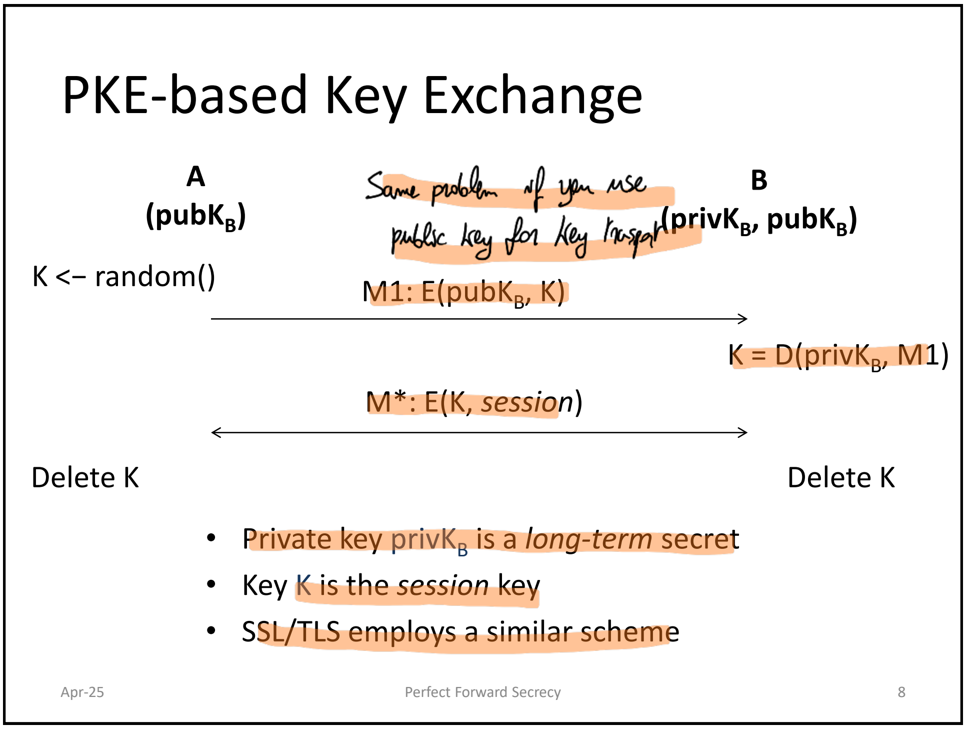


If K_{AB} is compromised, adversary just gets g^a, g^b, so they need to solve DLP.



PSK-DHE

- Ephemeral Diffie-Hellman
  - Keys a and b are ephemeral and one-time (per-session or per message)
  - Once a and b (and K) have been deleted there is no way to recover K, and thus the session, even if the long-term private K_{ab} is compromised
    - Neither A nor B can
    - The adversary has still to solve the DLP
  - K_{ab} is used for authentication, not for confidentiality anymore



PKE-based Key Exchange

A
($pubK_B$)

Same problem if you use public key for key transport

B
($privK_B$, $pubK_B$)

K <– random()

M1: E($pubK_B$, K) →

K = D($privK_B$, M1)

M*: E(K, *session*) ↔

Delete K

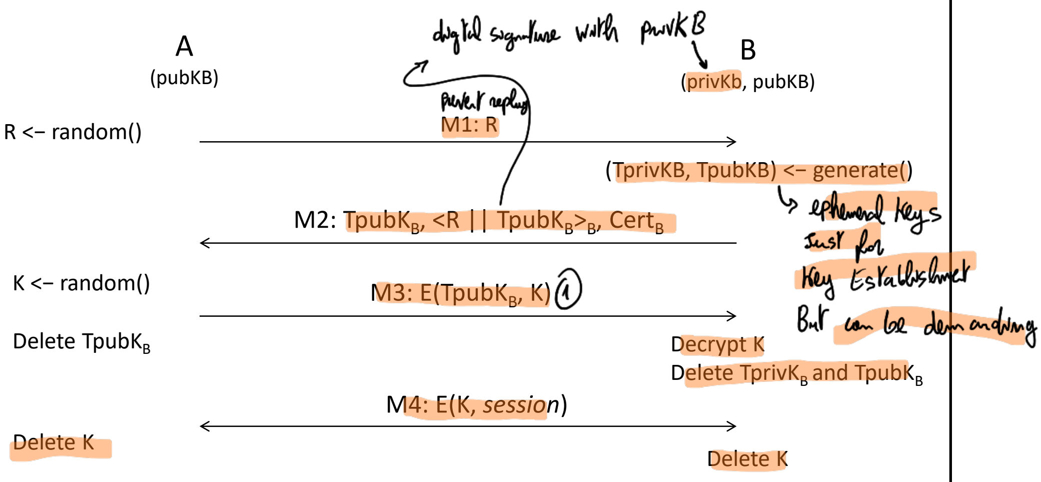
Delete K

- Private key $privK_B$ is a *long-term* secret
- Key K is the *session* key
- SSL/TLS employs a similar scheme

The problem

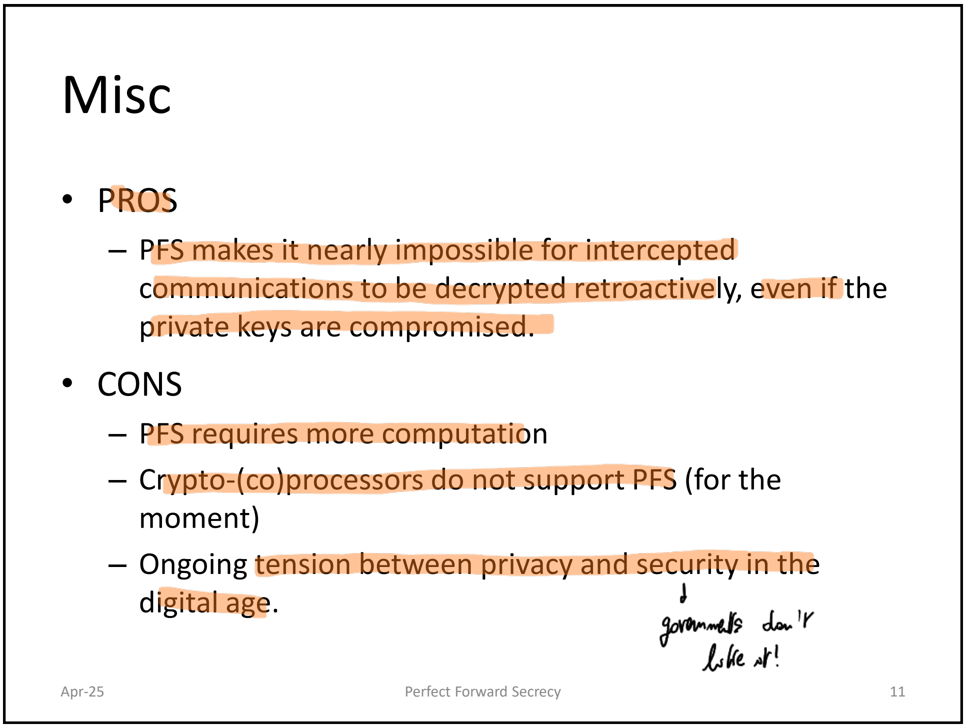
- The adversary records the encrypted session
- If the adversary compromises $privK_B$ then (s)he can recover K from CT
- Then, the adversary decrypts the session and violates secrecy
- The long-term secret becomes a single-point of failure

Ephemeral RSA (RSAE)


A
(pubKB)
digital signature with privKB
B
(privKb, pubKB)
prevent replay
R <– random()
M1: R
(TprivKB, TpubKB) <– generate()
ephemeral keys just for Key Establishment
But can be demanding
M2: $TpubK_B$, $<R \mid\mid TpubK_B>_B$, $Cert_B$
K <– random()
M3: $E(TpubK_B, K)$ (1)
Delete $TpubK_B$
Decrypt K
Delete $TprivK_B$ and $TpubK_B$
M4: E(K, *session*)
Delete K
Delete K

$<X>_B$: Bob's digital signature on X

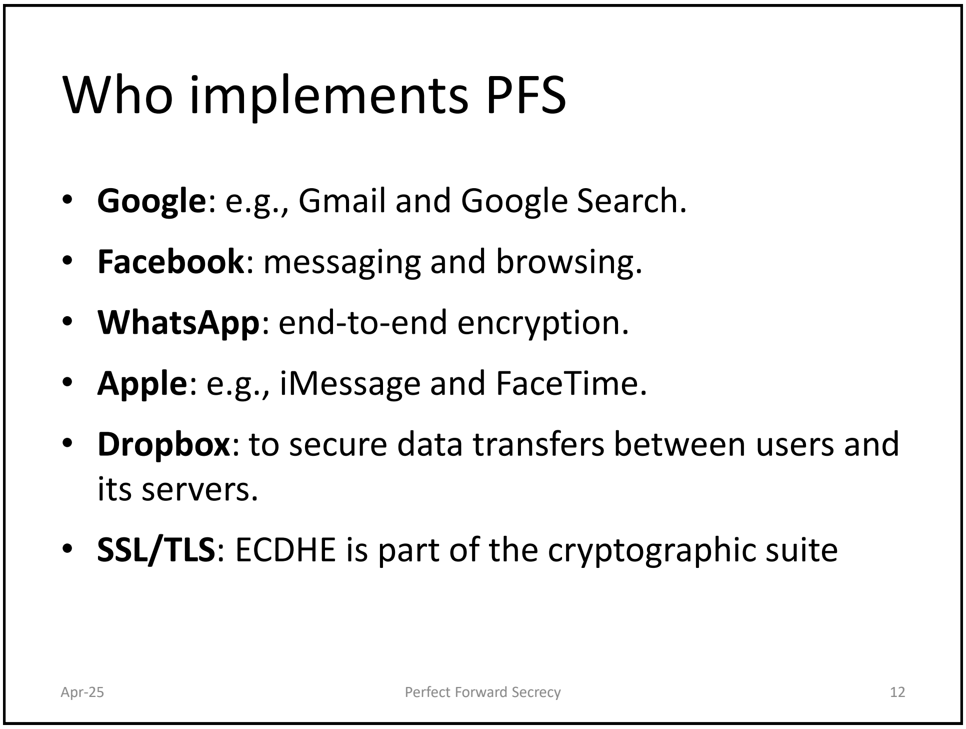
(1) If it is secure, it cannot be broken (Cyphertext only attack!)



Misc

- PROS
  - PFS makes it nearly impossible for intercepted communications to be decrypted retroactively, even if the private keys are compromised.
- CONS
  - PFS requires more computation
  - Crypto-(co)processors do not support PFS (for the moment)
  - Ongoing tension between privacy and security in the digital age.

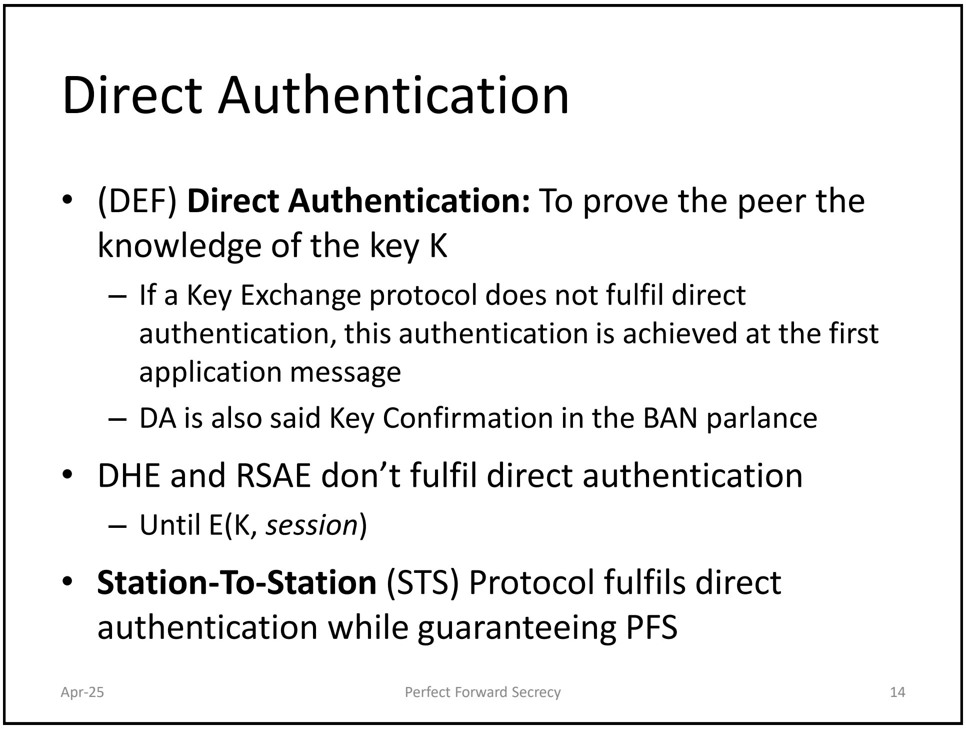
↓
governments don't like it!



Who implements PFS

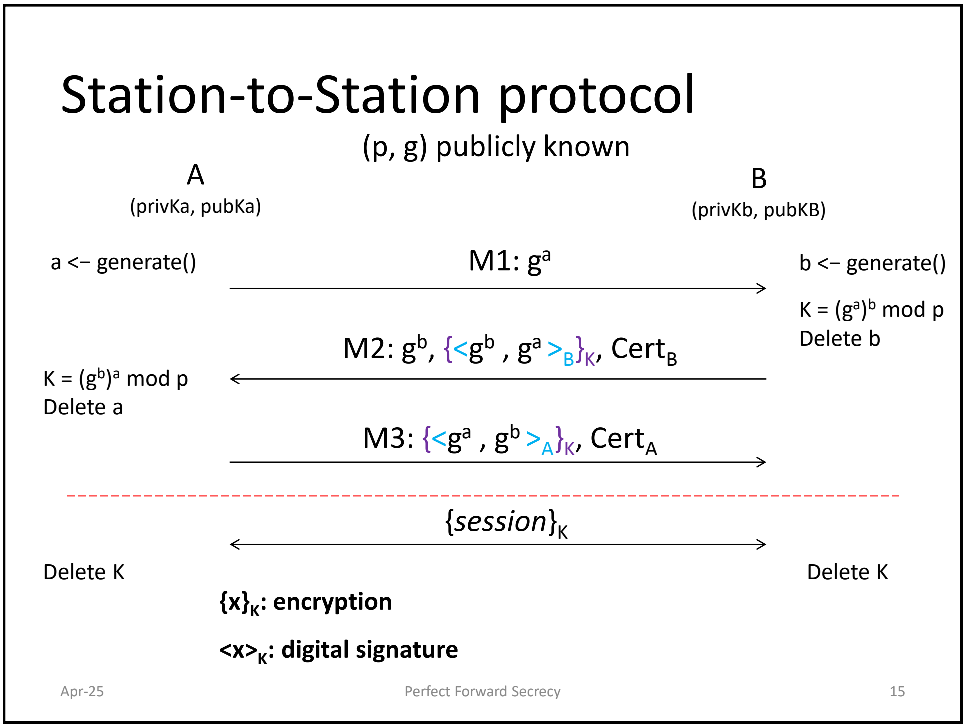
- **Google**: e.g., Gmail and Google Search.
- **Facebook**: messaging and browsing.
- **WhatsApp**: end-to-end encryption.
- **Apple**: e.g., iMessage and FaceTime.
- **Dropbox**: to secure data transfers between users and its servers.
- **SSL/TLS**: ECDHE is part of the cryptographic suite

DIRECT AUTHENTICATION



Direct Authentication

- (DEF) **Direct Authentication:** To prove the peer the knowledge of the key K
  - If a Key Exchange protocol does not fulfil direct authentication, this authentication is achieved at the first application message
  - DA is also said Key Confirmation in the BAN parlance
- DHE and RSAE don’t fulfil direct authentication
  - Until E(K, *session*)
- **Station-To-Station** (STS) Protocol fulfils direct authentication while guaranteeing PFS



Station-to-Station protocol

(p, g) publicly known

A (privKa, pubKa)

B (privKb, pubKB)

a <– generate()

M1: g^a →

b <– generate()

K = $(g^a)^b$ mod p

Delete b

M2: g^b, $\{<g^b, g^a>_B\}_K$, Cert_B ←

K = $(g^b)^a$ mod p

Delete a

M3: $\{<g^a, g^b>_A\}_K$, Cert_A →

$\{session\}_K$ ↔

Delete K

Delete K

$\{x\}_K$: encryption

$<x>_K$: digital signature

Apr-25 Perfect Forward Secrecy 15

15