Perfect Forward Secrecy

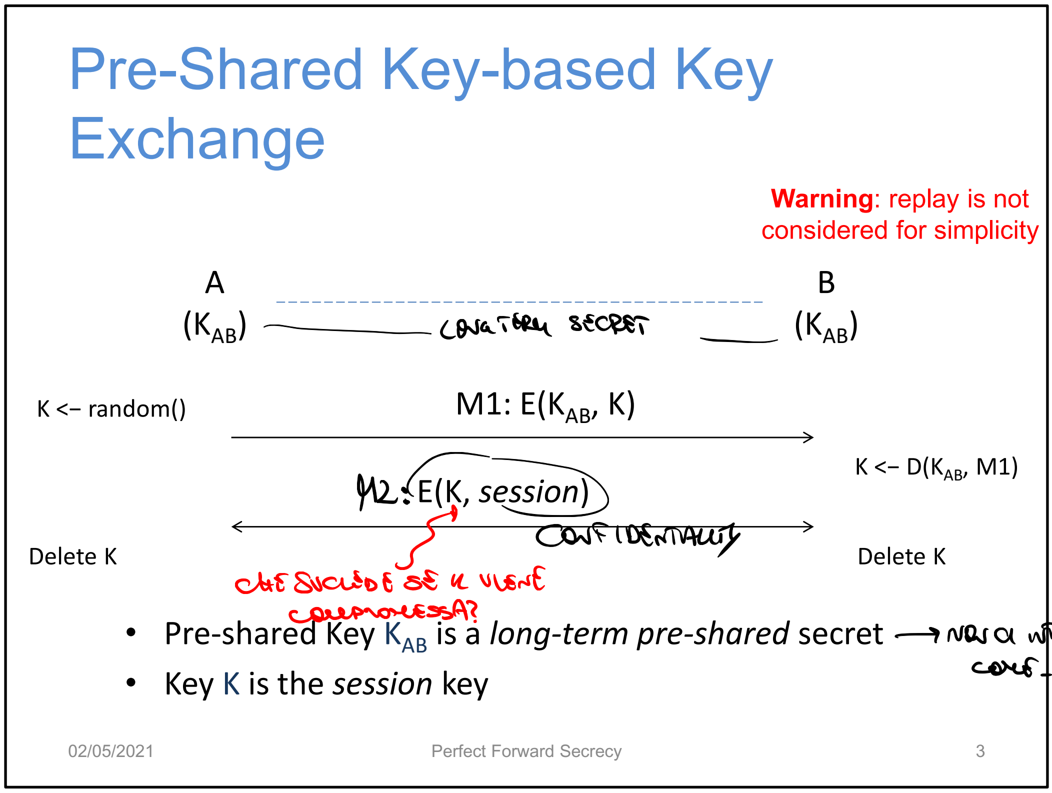
Gianluca Dini
Department of Ingegneria dell’Informazione
University of Pisa
Email: gianluca.dini@unipi.it

Version: 2021-05-02

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 for simplicity

A (K_{AB}) —— LONG TERM SECRET —— B (K_{AB})

K <– random()

M1: $E(K_{AB}, K)$ →

K <– $D(K_{AB}, M1)$

M2: $E(K, \textit{session})$ ↔ CONFIDENTIALITY

Delete K

Delete K

CHE SUCCEDE SE K VIENE COMPROMESSA?

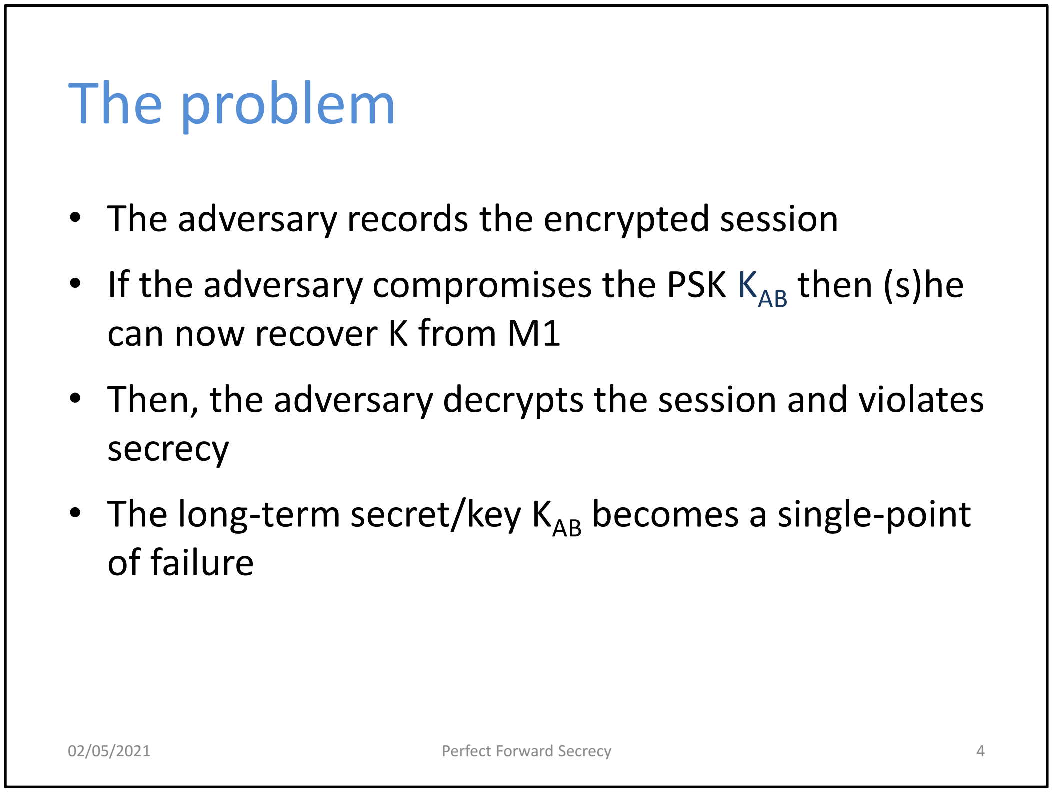
- Pre-shared Key K_{AB} is a *long-term pre-shared* secret → NON CI INTERESSA CONF.
- Key K is the *session* key

02/05/2021 Perfect Forward Secrecy 3

Let us consider a scenario where the long-term shared secret is a shared key K_{AB} (e.g., IpSec).

The long-term key KAB is used to exchange a shared key K. Actually, K_{AB} guarantees authenticity and confidentiality of K (here replay is not considered).

If the long-term key Kab is compromised, no future communication is secure. However, what about the past?

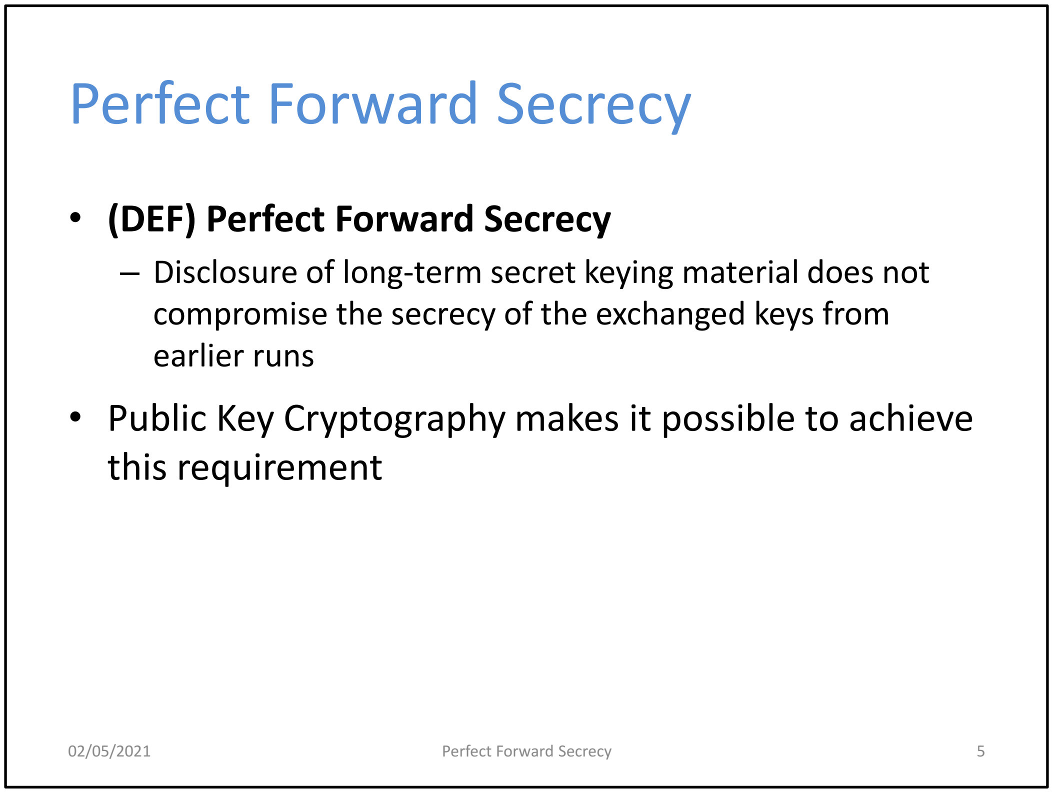


The problem

- 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

02/05/2021 Perfect Forward Secrecy 4

In case the long-term shared key is compromised, the previous sessions are compromised as well. Is there any trick that allows us to protect the past, even though we cannot protect the future?



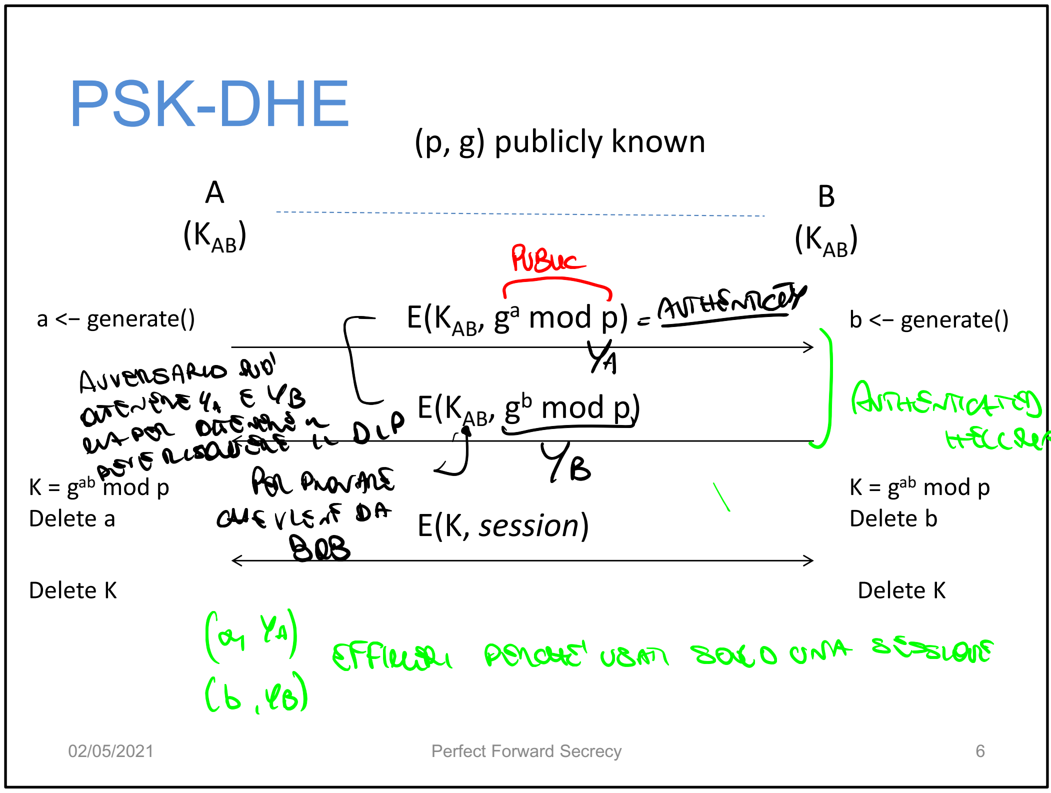
Perfect Forward Secrecy

- **(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

02/05/2021 Perfect Forward Secrecy 5

The PFS requirement definition

PFS can be achieved by means PKE, DH in particular.



PSK-DHE

(p, g) publicly known

A (K_{AB}) B (K_{AB})

a <– generate() b <– generate()

$E(K_{AB}, \overset{\text{PUBLIC}}{g^a \bmod p}) = \text{AUTHENTICITY}$ Y_A →

← $E(K_{AB}, g^b \bmod p)$ Y_B

AVVERSARIO PUÒ OTTENERE Y_A E Y_B MA PER OTTENERE K DEVE RISOLVERE IL DLP

PER PROVARE CHE VIENE DA BOB

AUTHENTICATED DIFFIE HELLMAN

K = g^{ab} mod p K = g^{ab} mod p
Delete a Delete b

$E(K, \textit{session})$ ↔

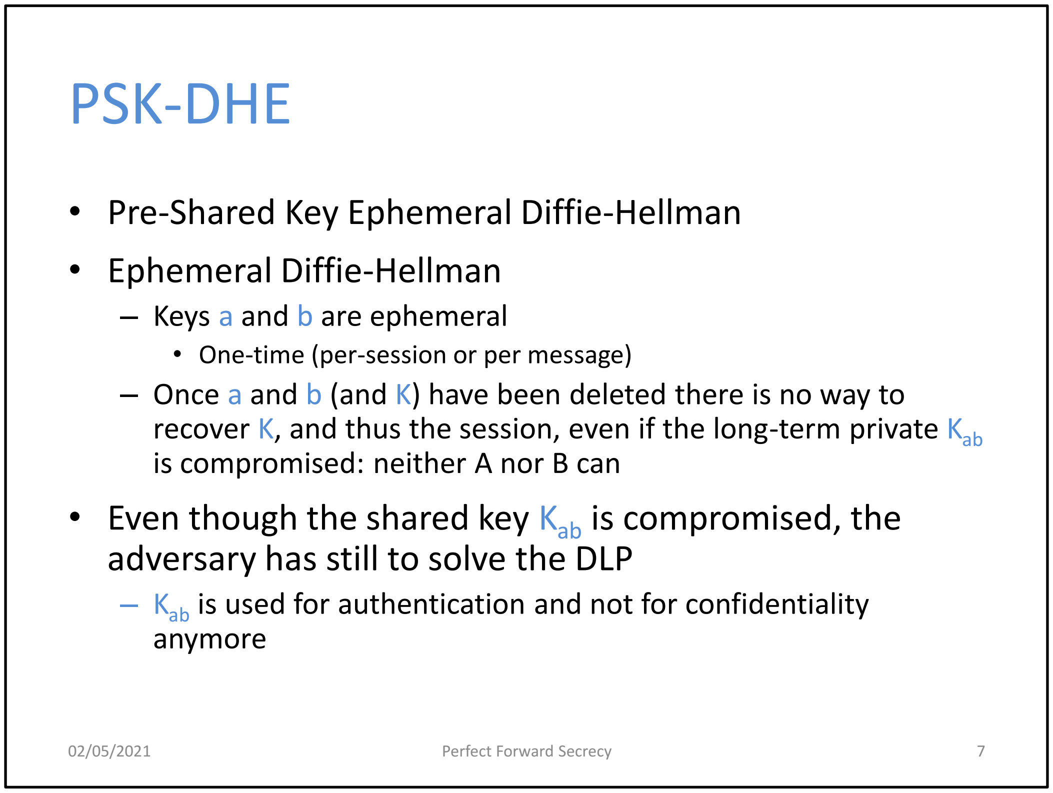
Delete K Delete K

(a, Y_A)
(b, Y_B) EFFIMERI PERCHE' USATI SOLO UNA SESSIONE

02/05/2021 Perfect Forward Secrecy 6

Pre-Shared Key – Diffie-Hellmann Encryption (PSK-DHE)

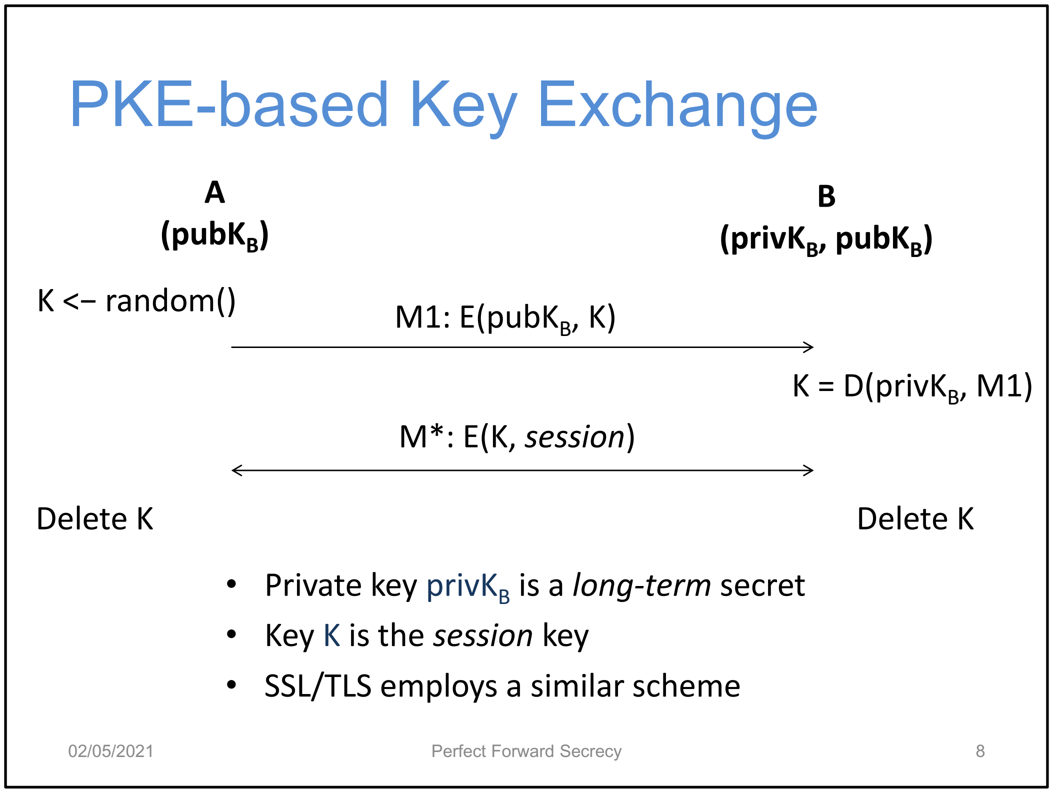
Let us consider a scenario where the long-term shared secret is a shared key K_{AB}. The long-term key KAB is used to authenticate the exchange if a shared key K by means of a DHKE. More precisely, K_{AB} guarantees authenticity of DH public keys. The important point to notice is that the DH secret keys (a and b) as well as the corresponding shared key K are *ephemeral*, i.e., they are freshly generate for the current execution of the protocol (current session). Private keys a and b are deleted as soon as the session key K is established. The session key K is deleted as soon ad the session is terminated. Now, if the long-term secret KAB is compromised, the adversary can only recover the DH public keys YA = g^a mod p and YB = g^b mod p. However, (s)he is unable to detect the shared key K unless (s)he is able to solve the DLP.



PSK-DHE

- Pre-Shared Key Ephemeral Diffie-Hellman
- Ephemeral Diffie-Hellman
  - Keys a and b are ephemeral
    - 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
- Even though the shared key K_{ab} is compromised, the adversary has still to solve the DLP
  - K_{ab} is used for authentication and not for confidentiality anymore

02/05/2021 Perfect Forward Secrecy 7



PKE-based Key Exchange

A **($pubK_B$)** **B** **($privK_B$, $pubK_B$)**

K <– random()

M1: E($pubK_B$, K) $\longrightarrow$

K = D($privK_B$, M1)

M*: E(K, *session*) $\longleftrightarrow$

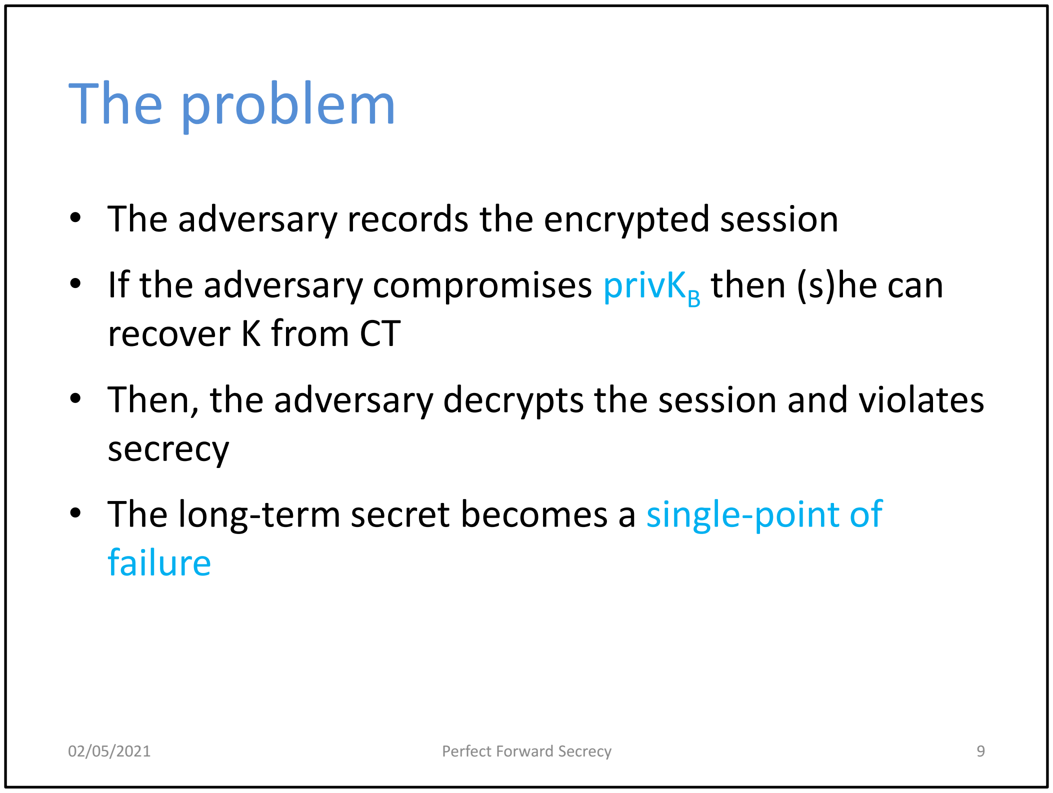
Delete K Delete K

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

02/05/2021 Perfect Forward Secrecy 8

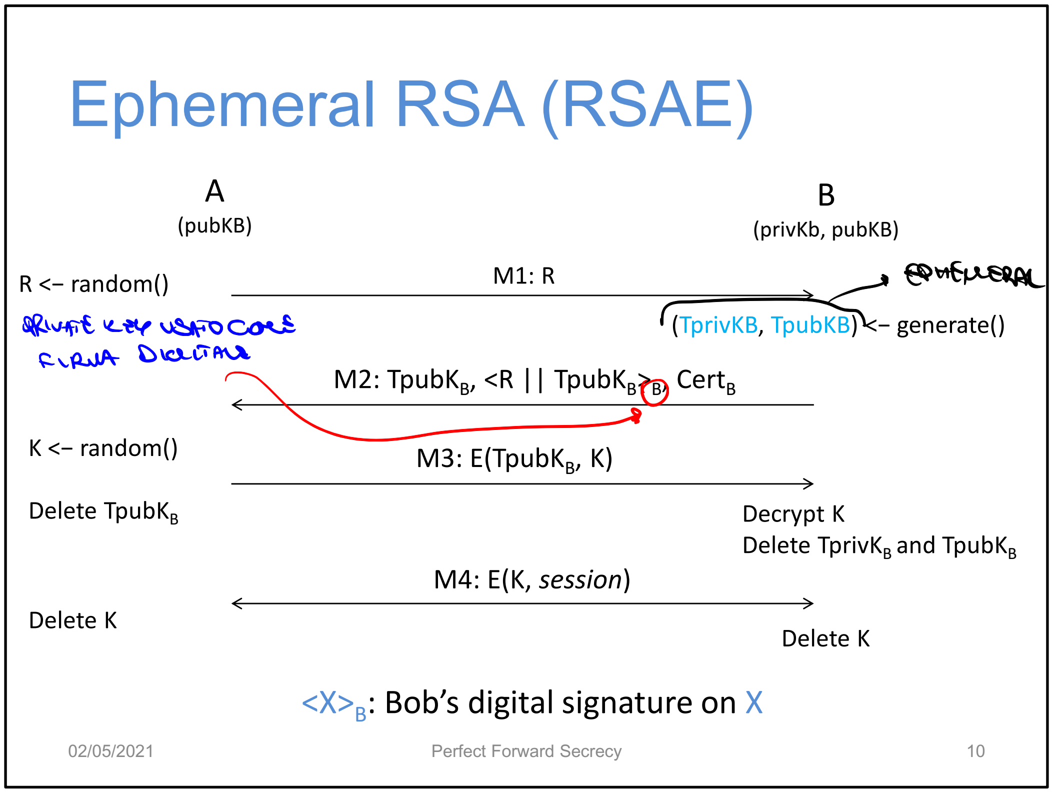
Let us consider a PKE-based key exchange. Alice generates a symmetric key and transmits it to B in its encrypted form under public key $pubK_B$. Bob decrypts the ciphertext by means of his private key $privK_B$. The session key k is used for all the session and then it is deleted. Bob's private key privKB is the long term secret.

If the session key K gets compromised, the whole session gets compromised. If the long term secret privKB gets compromised, future and past sessions are at risk. Actually, an adversary may recover any past session (s)he recorded the corresponding ciphertext.



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)

B (privKb, pubKB)

R <– random()

M1: R (A → B)

(TprivKB, TpubKB) <– generate()

EPHEMERAL KEYS

PRIVATE KEY USATO COME FIRMA DIGITALE

M2: $TpubK_B$, $<R \,||\, TpubK_B>_B$, $Cert_B$ (B → A)

K <– random()

M3: E($TpubK_B$, K) (A → B)

Delete $TpubK_B$

Decrypt K

Delete $TprivK_B$ and $TpubK_B$

M4: E(K, *session*) (A ↔ B)

Delete K

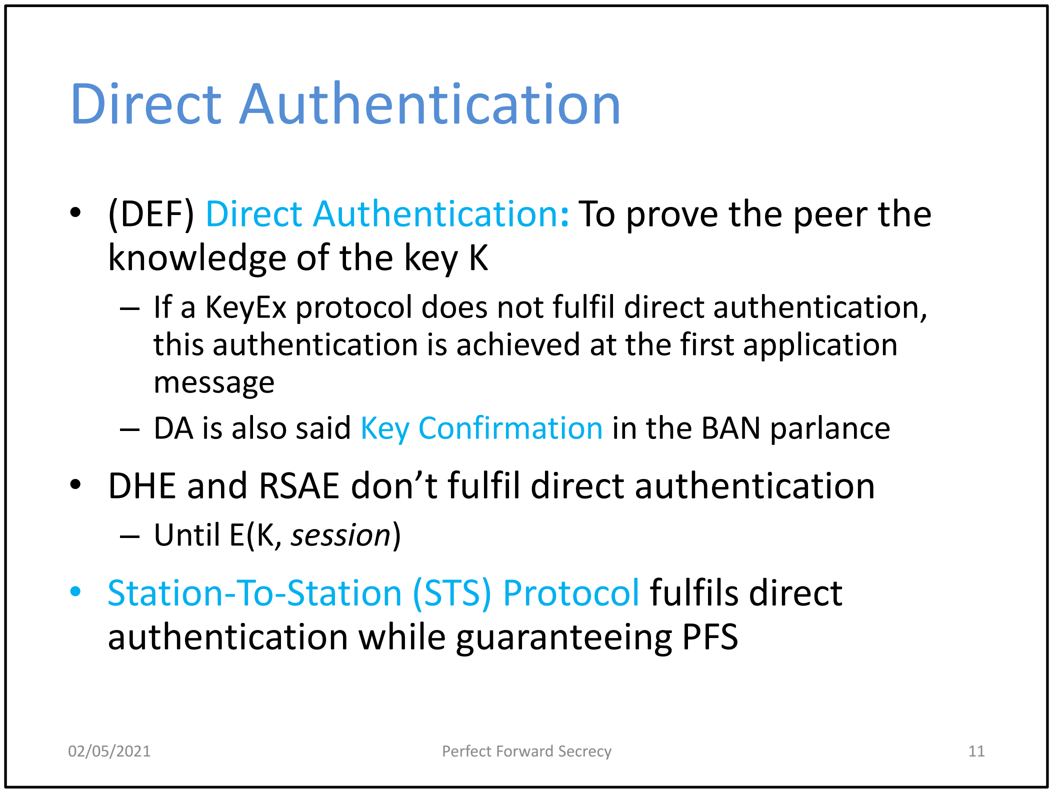
Delete K

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

02/05/2021 Perfect Forward Secrecy 10

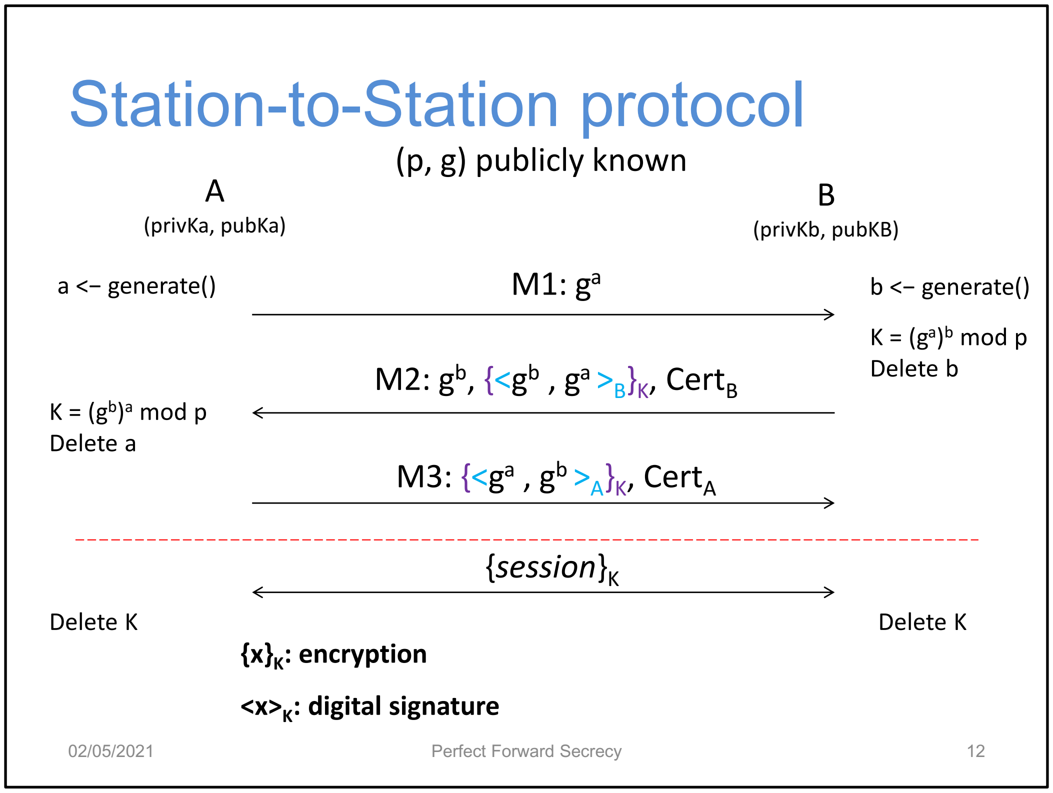
In the RSAE, Bob generates an *ephemeral* pair of private and public keys (TprivKB, TpubKB). Bob transmits Alice TpubKB. Bob authenticates this key by signing it with privKB. The random quantity R is used as a noce and makes it possible to avoid replay. Upon receiving TpubKB, Alice authenticates it by means of pubKB and then uses it to send Bob the session key K- Bob uses TpubKB to decrypt the session key K and then deletes that public key. When the session terminates, Alice and Bob delete the session key K.

If the session key K is compromised the corresponding session is compromised. If the long-term secret privKB is compromised, future sessions are at risk. However, past sessions are secure. Actually, an adversary can eavesdrop TpubKB but (s)he non able to determine TprivKB and thus decrypt the session key of previous sessions.



Direct Authentication

- (DEF) Direct Authentication: To prove the peer the knowledge of the key K
  - If a KeyEx 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

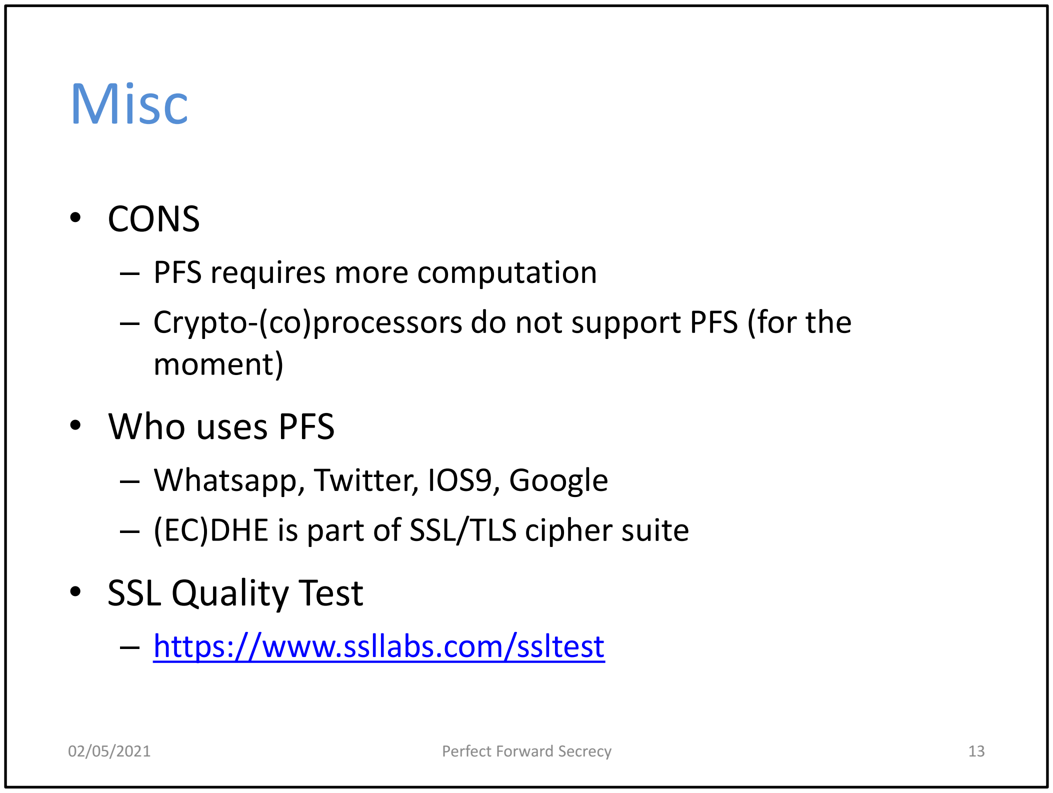
(p, g) publicly known

$\{x\}_K$: encryption

$<x>_K$: digital signature

02/05/2021 Perfect Forward Secrecy 12

Informally, the STS Protocol guarantees direct authentication because it uses the session K to encrypt messages M2 and M3. By means of these encryptions, peers prove each other that they hold the session key K.



Misc

- CONS
  - PFS requires more computation
  - Crypto-(co)processors do not support PFS (for the moment)
- Who uses PFS
  - Whatsapp, Twitter, IOS9, Google
  - (EC)DHE is part of SSL/TLS cipher suite
- SSL Quality Test
  - https://www.ssllabs.com/ssltest

02/05/2021 Perfect Forward Secrecy 13