



Aalto University

# Network Security: TLS 1.3 PSK and session resumption

Tuomas Aura, Aalto University

CS-E4300 Network security

# Outline

- Recall TLS 1.3 full handshake
- Pre-shared key (PSK) mode
- Session resumption

# TLS 1.3 full handshake

## Client

ClientHello

+ key\_share\*

+ signature\_algorithms\*

+ supported\_groups\*

+ server\_name\*

+ certificate\_authorities\*

4. Client authentication  
(typically omitted)

{Certificate\*}  
{CertificateVerify\*}  
{Finished}  
[Application data]

## Server

{encrypted}  
{encrypted}  
+ extension  
\* Optional

ServerHello

+ key\_share\*

{EncryptedExtensions}  
{CertificateRequest\*}  
{Certificate\*}  
{CertificateVerify\*}  
{Finished}  
[ApplicationData\*]

[Application data]

1. Parameter negotiation

2. DHE or ECDHE key exchange

3. Server authentication

5. Key confirmation

6. Protected session data

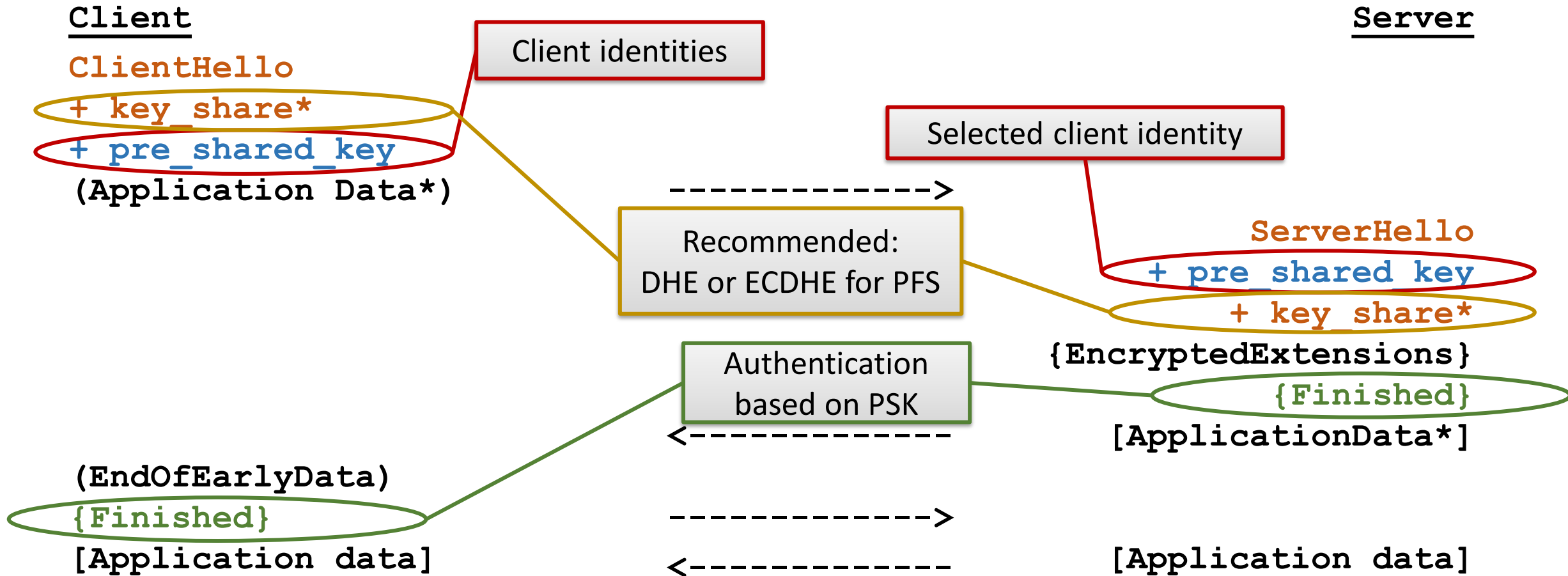
----->

<-----

----->

<----->

# Pre-shared key (PSK) mode



# Pre-shared key (PSK) mode

1. C  $\rightarrow$  S:  $N_C, g^x, \text{ClientIdentity}$
2. S  $\rightarrow$  C:  $N_S, g^y, \text{HMAC}_{K_{fks}}(\text{TH}),$   
early data
3. C  $\rightarrow$  S:  $\text{HMAC}_{K_{fkc}}(\text{TH})$

- **Mutual authentication** based on a **pre-established identity and session key (external PSK)**
  - **PSK** = pre-established shared key between C and S
  - HMAC keys  $K_{fks}$  and  $K_{fkc}$  for the Finished message are derived from PSK,  $g^{xy}$  and TH; and so are the session keys

# TLS 1.3 session resumption (1)

Client

Server

ClientHello

- + key\_share\*
- + signature\_algorithms\*
- + supported\_groups\*
- + server\_name\*
- + certificate\_authorities\*

----->

ServerHello

- + key\_share\*
- {EncryptedExtensions}
- {CertificateRequest\*}
- {Certificate\*}
- {CertificateVerify\*}
- {Finished}
- [ApplicationData\*]

Server packages the session state into an encrypted data blob called **session ticket** and sends it to the client

<-----

- {Certificate\*}
- {CertificateVerify\*}
- {Finished}

----->

<-----

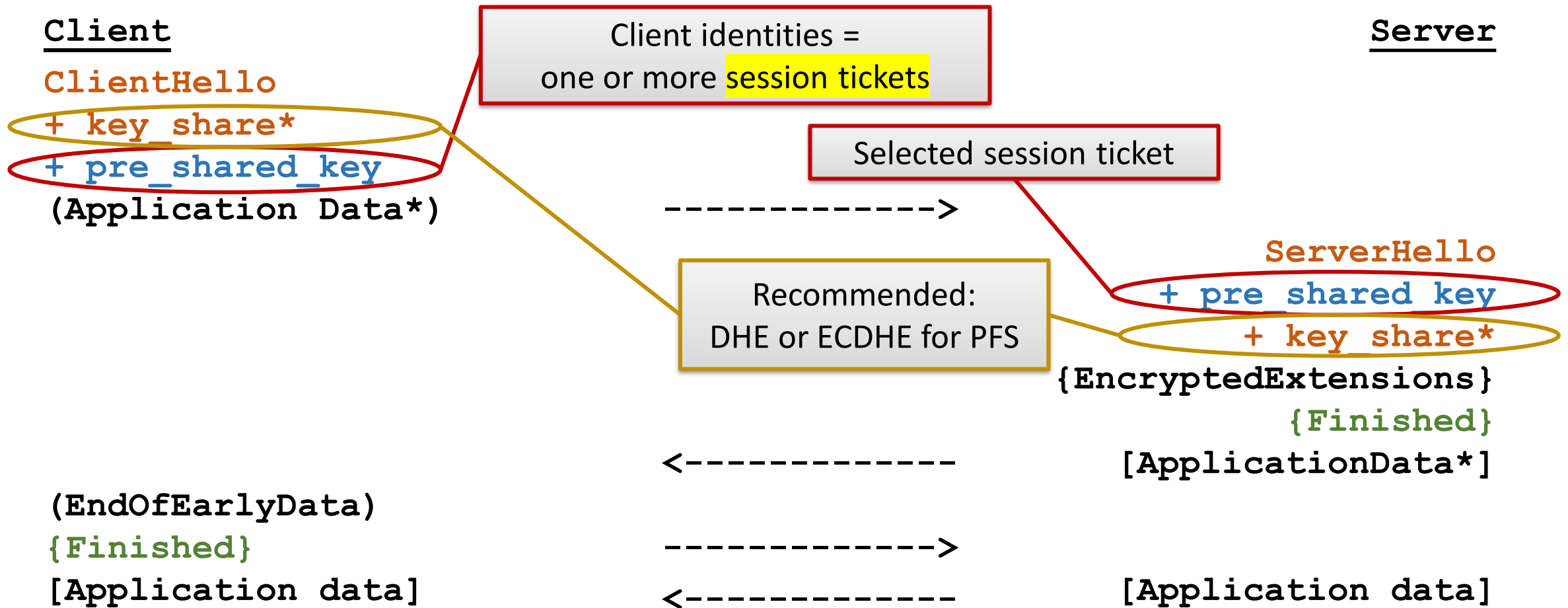
**NewSessionTicket**

[Application data]

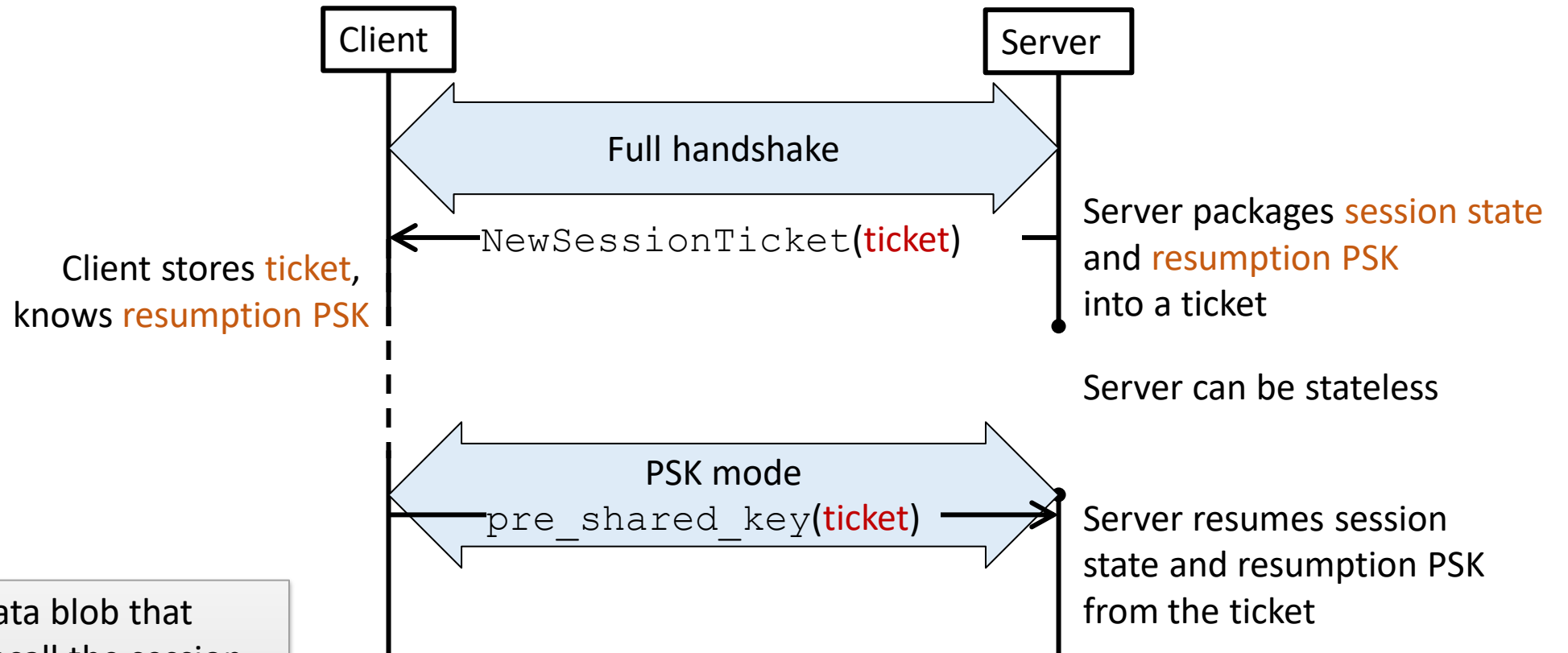
<----->

[Application data]

# TLS 1.3 session resumption (2)



# TLS 1.3 session resumption timeline



**Ticket** = opaque data blob that helps the server recall the session. Typically contains **encrypted session state** and **resumption PSK**. Only the server itself can decrypt the tickets that has created



# TLS 1.3 session resumption uses

- TLS 1.3 session resumption = PSK mode handshake with ticket as client identity and resumption key as the PSK
  - Currently the main purpose of the PSK mode
- When useful?
  - Server does not want to store the TLS sessions over idle periods
  - If client is authenticated with smartcard, avoids repeated user action
  - Mobile clients keep changing their IP address and need frequent reconnection
  - Resume the session with a different server instance in the cloud

# Key derivation

Inputs to key derivation:

1. PSK (external PSK or resumption PSK)
  2. DHE/ECDHE secret
  3. Transcript of handshake messages, up to the point where the key is derived
- } one or both, as available

Keys:

- client\_early\_traffic\_secret → used to derive AEAD keys for early data in 0-RTT (...)
- client/server\_handshake\_traffic\_secret → used to derive AEAD keys for handshake messages {...} and Finished HMAC keys
- client/server\_application\_traffic\_secret\_N → used to derive AEAD encryption keys for post-handshake application data and messages [...]
- resumption\_master\_secret and ticket\_nonce → derive resumption PSK
- exporter\_master\_secret → used to create keys for the application layer

# TLS 1.3 session resumption and identity

Client

ClientHello

+ key\_share\*

+ pre\_shared\_key

(Application Data\*)

Server

ServerHello

+ pre\_shared\_key

+ key\_share\*

{EncryptedExtensions}

{Finished}

[ApplicationData\*]

Server can refresh the ticket for PFS  
and for protecting client identity

NewSessionTicket

(EndOfEarlyData)

{Finished}

[Application data]

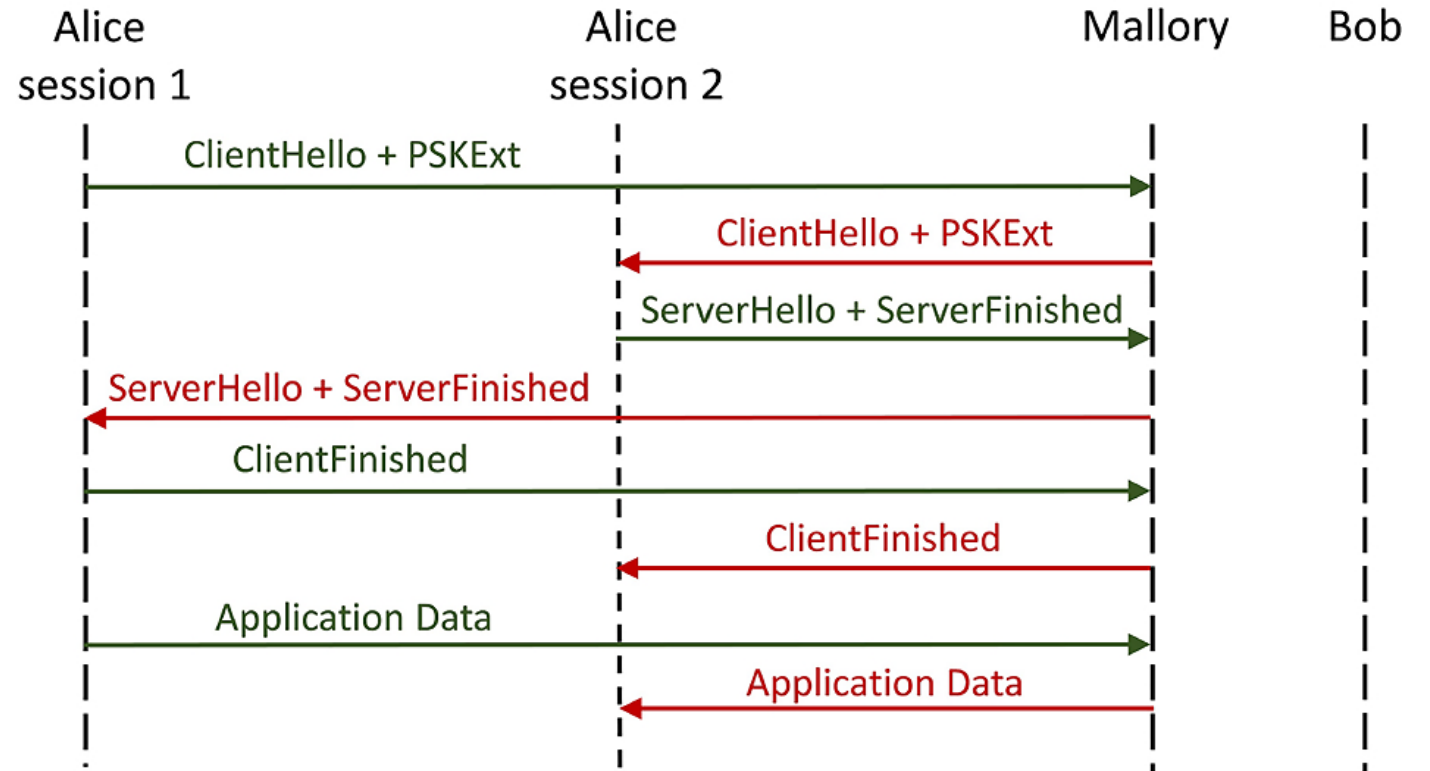
[Application data]

# Identity protection?

- Session tickets are encrypted
- Session ticket can become a pseudo-identifier
  - Server should regularly refresh ticket

# “Selfie attack”

- **Reflection attack** against external (out-of-band) PSK
  - Trick the client to connect to itself
  - Assumes the same entity can be both client and server
- PSK used mistakenly as a group key for two parties
  - Group key only authenticates the group, not the individual
- Solution: Use different PSK for each direction
  - For each PSK, Alice is either the client or server, never both for the same PSK



[Nir Drucker & Shay Gueron, Selfie: reflections on TLS 1.3 with PSK, 2019]