TLS supports session setup using a "pre-shared key" (so full handshake not needed):

\[
\begin{align*}
\text{client} & \quad \text{full handshake} \quad \text{server} \\
& \quad \nearrow \quad \searrow \\
& \quad \text{New SessionTicket (nonce, id)} \quad \Rightarrow \\
& \quad \text{pre-shared key} \quad \text{derived from session secrets, nonce, and id} \quad \leftarrow \\
\end{align*}
\]

Output of AKE protocol: \((\text{key}, \text{id})\)

- **Authenticity:** Only party that knows key is \text{id} (i.e., the party identified by id)
- **Secrecy:** All parties other than client and id cannot distinguish key from random (i.e., key is hidden)
- **Consistency:** If id also completes protocol, then it outputs \((\text{key}, \text{id})\)

\[
\begin{align*}
\text{client} & \quad \text{server} \\
& \quad \text{first message} \quad \text{vulnerable to replay attack} \\
& \quad \text{ClientHello + PreSharedKey(id)} \quad \leftarrow \text{"0-RTT data"} \\
& \quad \text{Enc}_{\text{AE}}(\text{k}, \text{data}) \quad \text{derived from pre-shared key} \\
& \quad \text{fresh key } k_{A-B}, k_{B-A} \text{ derived for rest of session (based on initial messages)} \\
\end{align*}
\]

\[
\begin{align*}
\text{client} & \quad \text{server} \\
& \quad \text{Vulnerable to replay attack} \\
& \quad \text{New Session Ticket} \quad \Rightarrow \\
& \quad \text{fresh key } k_{A-B}, k_{B-A} \text{ derived for rest of session (based on initial messages)} \\
\end{align*}
\]

Often also require forward secrecy: compromise of server in the future cannot affect secrecy of sessions in the past.

\[
\begin{align*}
& \text{In TLS, server secret is a signing key} - \text{fresh Diffie-Hellman secret used for each session is fresh ("ephemeral")} \\
& \text{Compromising signing key allows impersonation of server, but does not break secrecy of past sessions} \\
& \text{As we will see, not all AKE protocols provide forward secrecy} \\
\end{align*}
\]

Very tricky to get right as we will see... Just use TLS!

**AKE from PKI:** suppose server has certificate authenticating a public key for a PKE scheme (CCA-secure):

\[
\begin{align*}
\text{Alice} & \quad \text{Alice} \quad \text{Bank} \\
& \quad \text{Alice} & \quad \text{Bank} \\
& \quad k \leftarrow k & \quad \text{sk}_{\text{Bank}} \leftarrow \text{sk}_{\text{Bank}} \\
& \quad \text{c} \leftarrow \text{Enc}(\text{sk}_{\text{Bank}}, (r, k)) & \quad \text{sk}_{\text{Bank}} \leftarrow \text{sk}_{\text{Bank}} \\
& \quad \text{r, cert}_{\text{Bank}} & \quad \text{cert}_{\text{Bank}} \\
& \quad \text{Alice} & \quad \text{Alice} \\
& \quad k & \quad \text{Alice} \\
& \quad \text{If we do not encrypt the nonce } r \text{, replay attack possible (adversary replays messages from past session - e.g., "send Eve } f(r) \text{")} \\
\end{align*}
\]

\[
\begin{align*}
& \text{If we do not encrypt the nonce } r \text{, replay attack possible (adversary replays messages from past session - e.g., "send Eve } f(r) \text{")} \\
& \text{c} \text{ nonce ensures freshness} \\
\end{align*}
\]
Mutual authentication: Bank has certificate identifying public key for PKE scheme.
Alice has certificate identifying public key for signature scheme.

\[ k^2 \rightarrow \text{Alice} \]
\[ r, \text{cert}_{\text{Bank}} \]
\[ c \leftarrow \text{Enc}(\text{pk}_{\text{Bank}}, (k, "Alice")) \]
\[ \sigma \leftarrow \text{Sign}(\text{sk}_{\text{Alice}}, (r, c, "Bank")) \]
\[ (k, \text{Alice}) \leftarrow \text{Dec}(\text{sk}_{\text{Bank}}, c) \]
\[ \text{check Alice matches id in certificate} \]
\[ k, \text{Alice} \]
\[ \text{check Alice's signature on } (r, c, "Bank") \text{ under pk}_{\text{Bank}} \text{ in cert}_{\text{Alice}} \]

Above protocol provides static (no forward secrecy) mutual authentication.

Most variants to this protocol are broken! AKE very delicate:
- Example: Suppose Alice encrypts \((k, r)\) instead of \((k, "Alice")\) like in the server-auth protocol above.
- Vulnerable to "identity misbinding" attack where Alice thinks she's talking to Bank but Bank thinks it's talking to Eve.

\[ k^2 \rightarrow \text{Alice} \]
\[ r, \text{cert}_{\text{Bank}} \]
\[ c \leftarrow \text{Enc}(\text{pk}_{\text{Bank}}, (k, r)) \]
\[ \sigma \leftarrow \text{Sign}(\text{sk}_{\text{Alice}}, (r, c, "Bank")) \]
\[ (k, \text{Alice}) \leftarrow \text{Dec}(\text{sk}_{\text{Bank}}, c) \]
\[ \text{check Alice's signature on } (r, c, "Bank") \text{ under pk}_{\text{Bank}} \text{ in cert}_{\text{Alice}} \]
\[ \Rightarrow \text{Bank thinks it's talking to Eve} \]

if Alice now sends "deposit this check into my account" to Bank:
Bank deposits it into Eve's account!  
observe that Eve did not break secrecy (she does not know \(k\)), but nevertheless broke consistency.

Above protocols supported by TLS 1.2, but deprecated in TLS 1.3 due to lack of forward secrecy.

To get forward secrecy, use ephemeral keys:

\[ k^2 \rightarrow \text{Alice} \]
\[ k, \text{Bank} \]
\[ \text{cert}_{\text{Bank}}, \sigma \leftarrow \text{Sign}(\text{sk}_{\text{Bank}}, pk) \]
\[ c \leftarrow \text{Enc}(sk, k) \]
\[ \text{for signature scheme} \]
\[ \text{sk}_{\text{Bank}}, \text{cert}_{\text{Bank}} \]
\[ k \leftarrow \text{Dec}(sk, c) \]
\[ k, L \]
\[ \text{for signature scheme} \]
\[ \text{_bank} \]

Provides one-sided authentication:
- Signature binds \(pk\) to Bank.
- Forward secure since each \(pk\) used only once.
- Long-term secret is signing key.

hardware security module (used to protect cryptographic secrets)

Problem: Does not provide "HSM security".
- Suppose adversary breaks into the bank and learns a single \((pk', sk')\) pair with \(\sigma \leftarrow \text{Sign}(sk_{\text{Bank}}, pk)\)
- Adversary can now impersonate the bank to any client:
  - adversary always use the message \((pk', \text{cert}_{\text{Bank}}, \sigma)\)
  - defending against this requires freshness from client
- can decrypt keys for all clients that responds!

\[ \text{totally broken without signature,} \]
\[ \text{adversary can replace pk with pk' and} \]
\[ \text{learn Alice's chosen key} \]
Alice \xrightarrow{pk} c \xleftarrow{c = \text{Enc}(pk, k)} \text{Bank} {k' = k}

\text{cert}_{\text{Bank}} \xrightarrow{\sigma = \text{Sig}(\text{sk}_{\text{Bank}}, (pk, k))} \text{Bank} \xrightarrow{k, L}

\text{Provides \text{HMAC} security: client chooses fresh \( pk \) each time, so signature on \( pk \) functions as a "proof" that the other party possesses signing key for \( \text{id} \) identified by \text{cert}_{\text{Bank}}\}

In many cases, also want to hide the endpoint (the \( \text{id} \) identified by \text{cert})

Possible by encrypting two keys \((k, k')\) and using \( k' \) to encrypt \text{cert}_{\text{Bank}}

\text{Diffie-Hellman key-exchange: substitute Diffie-Hellman handshake for the PKE scheme (simpler)}

(TLS 1.2, 1.3)