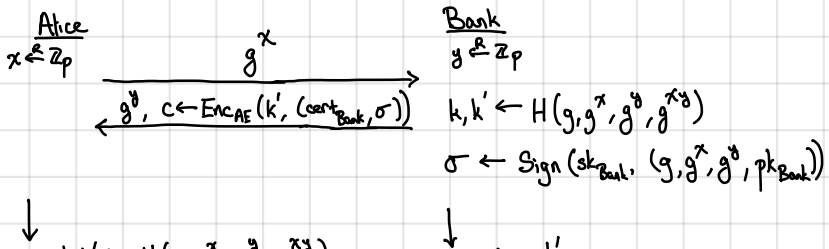
Basic flow of Diffie-Hellman based AKE:

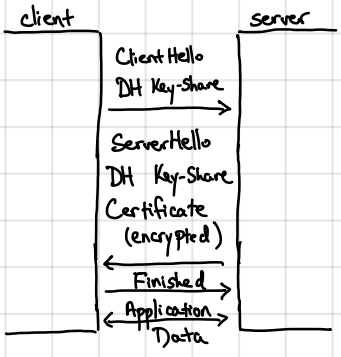


Intuition: $cert_{Bank}$ identifies server as Bank (with pk_{Bank})
σ binds the session parameters (g, g^x, g^y) to the public key identified by $cert_{Bank}$

End of protocol: Alice knows she is talking to Bank (but not vice versa!)

"one-sided AKE" — most common mode on the web

→ Basis of TLS 1.3 handshake ("one-sided" AKE) ALWAYS USE TLS 1.3 – Don't invent your own AKE protocol!



ClientHello: List of supported ciphersuites
(e.g., AES-GCM-128, AES-GCM-256)
Possible TLS extensions

ServerHello: Chosen ciphersuite

older systems / foreign systems may prefer different ciphers

older versions of TLS vulnerable to cipher downgrade attacks

Application layer secured using unidirection keys $k_{A \to B}$ and $k_{B \to A}$

Protecting signing keys is extremely important for a CA

Common approach: threshold signatures

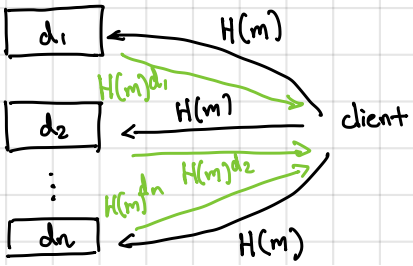
Recall RSA signatures: $\sigma = H(m)^d \pmod N$

Idea: Split signing key d into many "shares":

Sample $d_1, \ldots, d_n \xleftarrow{R} \mathbb{Z}_{\varphi(N)}$ such that $d_1 + d_2 + \cdots + d_n = d$

Observe: given any subset of shares (that is not the full set), d is perfectly hidden

Suppose we give one share d_i to each server



given signature shares $\sigma_i = H(m)^{d_i}$,

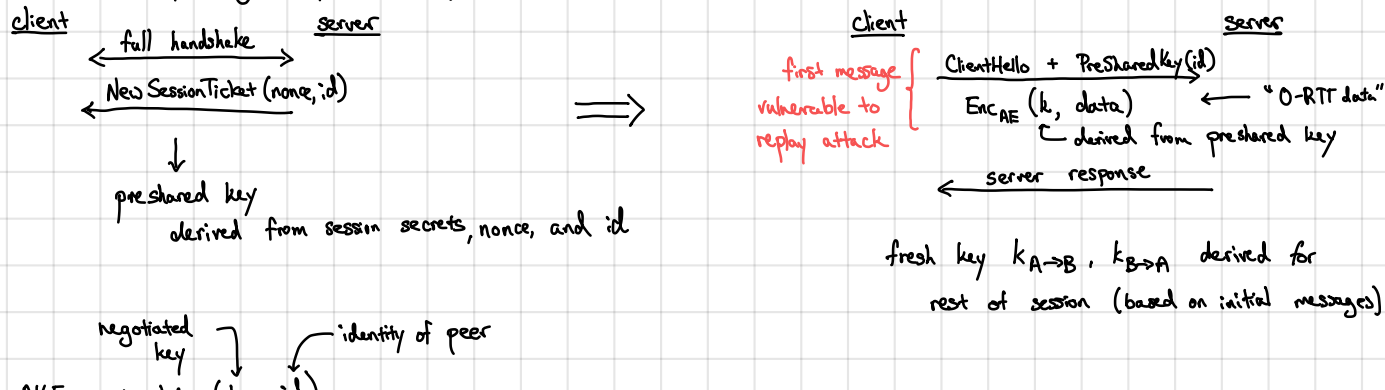
let $\sigma = \prod_{i \in [n]} \sigma_i$

$= \prod_{i \in [n]} H(m)^{d_i}$

$= H(m)^{\sum_{i \in [n]} d_i} = H(m)^d$

which is a signature on m

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



Output of AKE protocol: (key, id) — key: negotiated key; id: identity of peer

Authenticity: Only party that knows key is id (i.e., the party identified by id)

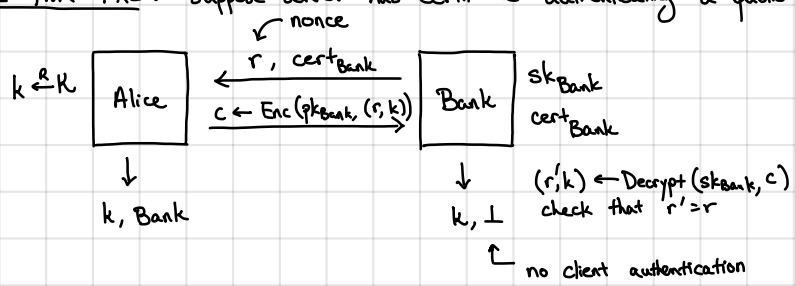
Secrecy: All parties other than client and id cannot distinguish key from random (i.e., key is hidden)

Consistency: If id also completes protocol, then it outputs (key, id_{client})
↳ if we do not have client authentication, then id_{client} is empty

Often also require forward secrecy: compromise of server in the future cannot affect secrecy of sessions in the past
↳ In TLS, server secret is a signing key — fresh Diffie-Hellman secret used for each session is fresh ("ephemeral")
Compromising signing key allows impersonation of server, but does not break secrecy of past sessions
↳ As we will see, not all AKE protocols provide forward secrecy

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

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



Yields statically-secure AKE (no forward secrecy)

Compromise of sk_{Bank} compromises all past sessions

If we do not encrypt the nonce r: replay attack possible (adversary replays messages from past session — e.g., "send Eve $10")
↳ nonce ensures freshness