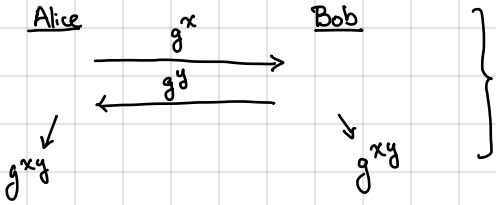
Now that we have digital signatures, let's revisit the question of key exchange (with active security)



} completely vulnerable to an active network adversary that can intercept and inject packets

In addition, should guarantee that one compromised session should not affect other honest sessions
- Alice ↔ Eve should not compromise security of Alice ↔ Bob

Authenticated key exchange (AKE): provides security against active adversaries
- Requires a "root of trust" (certificate authority) → we need some binding between keys and identities



(one-time setup, at least for duration of validity period)

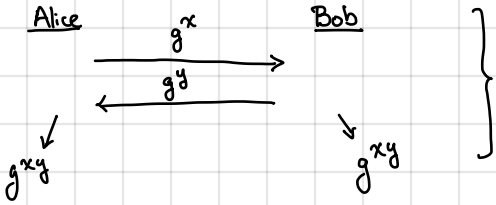
↑ the certificate binds Alice's public key pk_{Alice} to Alice's identity

- Certificates typically have the following format (X509):
  - Subject (entity being authenticated)
  - Public key (public key for subject for signature scheme)
  - CA: identity of the CA issuing the certificate
  - Validity dates for certificate
  - CA's signature on certificate

← the browser and operating system have a set of hard-coded certificate authorities and their respective public keys (usually several hundred authorities)

[public-key infrastructure (PKI)]

Now that we have digital signatures, let's revisit the question of key exchange (with active security)



} completely vulnerable to an active network adversary that can intercept and inject packets

In addition, should guarantee that one compromised session should not affect other honest sessions
- Alice ↔ Eve should not compromise security of Alice ↔ Bob

Authenticated key exchange (AKE): provides security against active adversaries
- Requires a "root of trust" (certificate authority) → we need some binding between keys and identities



(one-time setup, at least for duration of validity period)

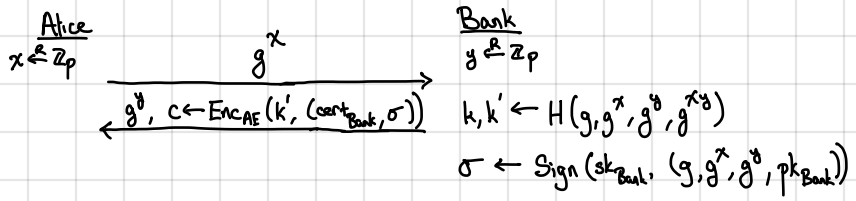
↑ the certificate binds Alice's public key pk_{Alice} to Alice's identity

- Certificates typically have the following format (X509):
  - Subject (entity being authenticated)
  - Public key (public key for subject for signature scheme)
  - CA: identity of the CA issuing the certificate
  - Validity dates for certificate
  - CA's signature on certificate

← the browser and operating system have a set of hard-coded certificate authorities and their respective public keys (usually several hundred authorities)

[public-key infrastructure (PKI)]

Basic flow of Diffie-Hellman based AKE:



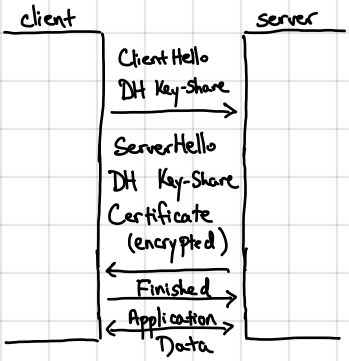
derive $k, k' \leftarrow H(g, g^x, g^y, g^{xy})$
check σ is signature on (g, g^x, g^y, pk_{Bank})
under pk_{Bank} is the public key identified by $cert_{Bank}$

session key k'

} 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

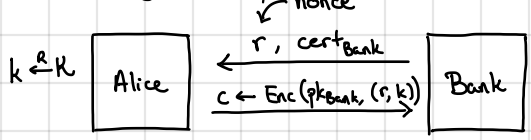
older versions of TLS vulnerable to cipher downgrade attacks

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

In TLS 1.3, the only long-term secret on the server is a signing key. This is critical for achieving forward secrecy.

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

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



sk_{Bank}
$cert_{Bank}$

$(r', k) \leftarrow Decrypt(sk_{Bank}, c)$
check that $r' = r$

Yields statically-secure authenticated key agreement (no forward secrecy)
Compromise of sk_{Bank} compromises all past sessions