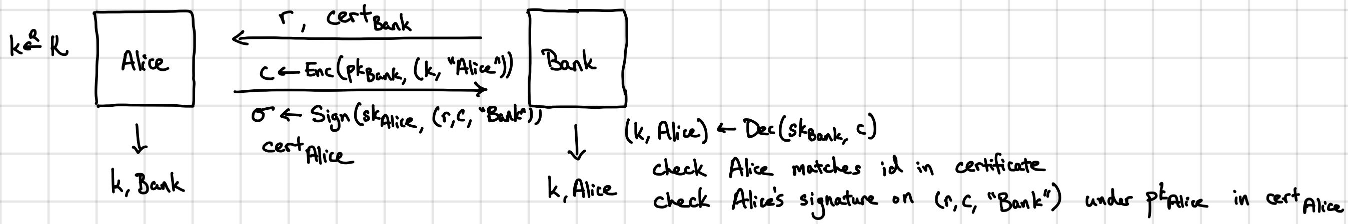
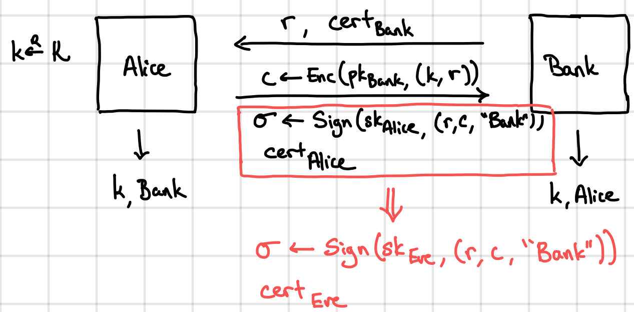
Mutual authentication: Bank has certificate identifying public key for PKE scheme
Alice has certificate identifying public key for signature scheme



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 $(k, \text{"Alice"})$ 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:

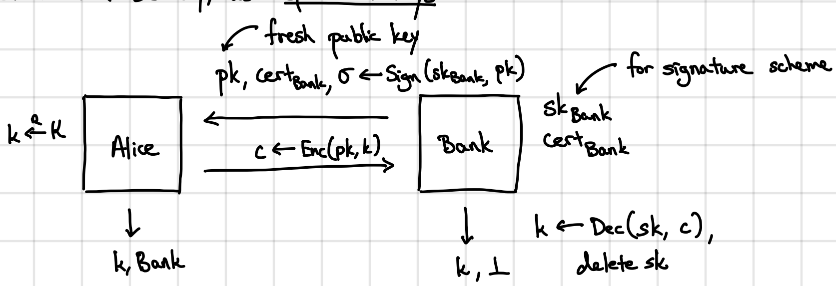


$\Rightarrow$ Bank thinks it's talking to Eve

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:



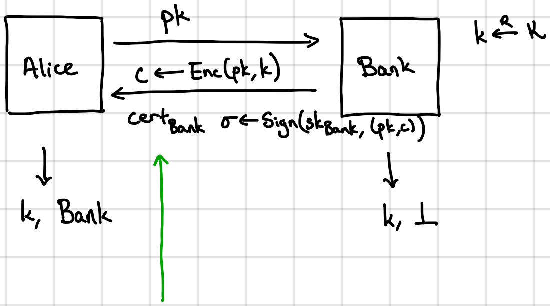
Provides one-sided authentication (signature binds pk to Bank)
Forward secure since each pk used only once and long-term secret is signing key

totally broken without signature, adversary can replace pk with pk' and learn Alice's chosen key

Problem: Does not provide "HSM security" (hardware security module (used to protect cryptographic secrets))

- ↳ Suppose adversary breaks into the bank and learns a single (pk', sk') pair with $\sigma \leftarrow Sign(sk_{Bank}, pk')$
- ↳ Adversary can now impersonate the bank to any client:
  - adversary always use the message $(pk', cert_{Bank}, \sigma)$
    - ↳ can decrypt keys for all clients that responds!

} defending against this requires freshness from client



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

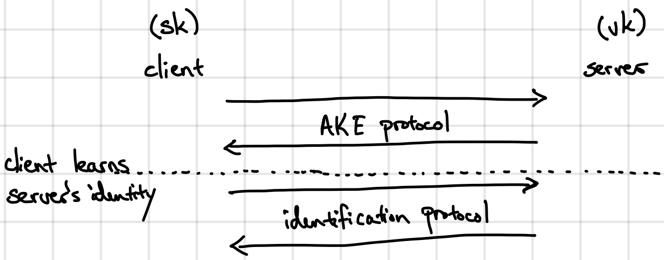
In many cases, also want to hide the endpoint (the id identified by cert)
Possible by encrypting two keys (k, k') and using k' to encrypt $\text{cert}_{\text{Bank}}$

Diffie-Hellman key-exchange: substitute Diffie-Hellman handshake for the PKE scheme (simpler)
(TLS 1.2, 1.3)

TLS 1.3 and authenticated key-exchange protocols on the Internet typically provide <u>one-sided</u> authentication (i.e., client learns id of the server, but not vice versa)

<u>Question</u>: how does the client authenticate to the server (<u>without</u> providing a certificate)
↳ e.g., how does client login to a web service?

<u>Typical setting</u>: client and server assumed to have some shared state [e.g., client has a password and server has an HMAC of the password]

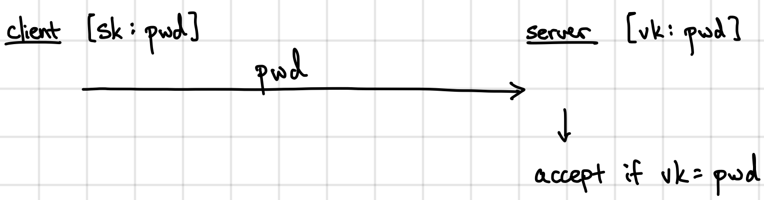


} <u>cannot</u> replace this with anonymous key exchange
↳ becomes vulnerable to a man-in-the-middle attack

<u>Threat models</u>: Adversary's goal is to authenticate to server

- <u>Direct attack</u>: adversary only sees vk and needs to authenticate
  (e.g., physical analogy: door lock — adversary can observe the lock, does not see the key sk)
- <u>Eavesdropping attack</u>: adversary gets to <u>observe</u> multiple interactions between honest client and the server
  (e.g., physical analogy: wireless car key — adversary observes communication between car key and car)
- <u>Active attack</u>: adversary can impersonate the server and interact with the honest client
  (e.g., physical analogy: fake ATM in the mall — honest clients interact directly with the adversary)

Simple (insecure) password-based protocol:



Not secure even against direct attacks! Adversary who learns vk can authenticate as the client [adversary who breaks into server learns user's password!]

NEVER STORE PASSWORDS IN THE CLEAR!

<u>Slightly better solution</u>: hash the passwords before storing

server maintains mappings
Alice ↦ $H(pwd_{Alice})$
Bob ↦ $H(pwd_{Bob})$

where H is a collision-resistant hash function