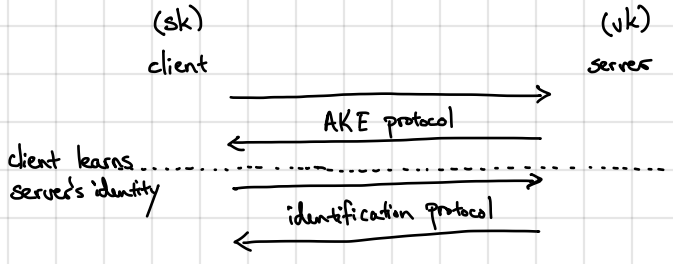
TLS 1.3 and authenticated key-exchange protocols on the Internet typically provide <u>one-sided</u> authentication (i.e., client learns id of the server, but not vice versa)

<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)

<u>Basic approach</u>: password-based authentication — only secure against direct attacks

A protocol secure against eavesdropping — needs a <u>fresh</u> message every authentication

Schnorr's protocol based on discrete log (no secrets, stateless, interactive)

client ($x \in \mathbb{Z}_p$) server ($h = g^x$)

"commitment" $u = g^r$

"challenge" $t \xleftarrow{R} \mathbb{Z}_p$

"response" $z = r + xt$

check $g^z = u \cdot h^t$

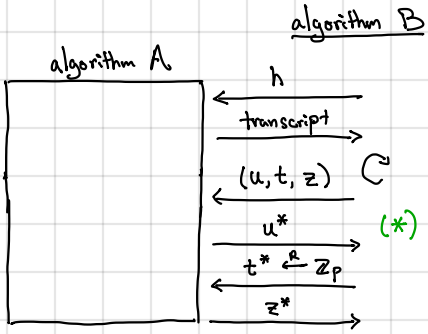
client will prove knowledge of x to the server

Protocols with this structure called a Σ-protocol (also require that verifier's challenge is a random string)

Correctness: $g^z = g^{r+xt} = g^r (g^x)^t = u \cdot h^t$

Security against passive adversaries: Suppose A can break security with probability 1: (will relax later)

- Algorithm A can request authentication transcripts, so reduction must simulate these



This is distributed identically to the real protocol

- in real protocol, $t \xleftarrow{R} \mathbb{Z}_p$, $z = r + tx$ where $r \xleftarrow{R} \mathbb{Z}_p$

given z and t, $u = g^z / h^t$ is fully determined

To extract, algorithm B will "reset" state of algorith A to (*)

- Namely, algorithm B runs A to get a transcript (u^*, t_1^*, z_1^*)
- Then algorithm B "rewinds" A to (*) but supplies a different $t_2^* \xleftarrow{R} \mathbb{Z}_p$
- Let (u^*, t_2^*, z_2^*) be the resulting transcript
- Algorithm B outputs $x = (z_1^* - z_2^*)(t_1^* - t_2^*)^{-1}$

if A succeeds w.p. 1, then

$g^{z_1^*} = u^* \cdot h^{t_1^*}$

$g^{z_2^*} = u^* \cdot h^{t_2^*}$

$\Downarrow$

$g^{z_1^* - z_2^*} = h^{t_1^* - t_2^*}$

$\Downarrow$

$z_1^* - z_2^* = x(t_1^* - t_2^*)$

In general, if A succeeds w.p. ε, then algorithm B succeeds w.p. $\varepsilon^2 - \varepsilon/p$

["rewinding lemma"]

We refer to this property as a "proof of knowledge"

↳ Any client that succeeds in this protocol with good probability must in fact know x.

Is this protocol secure against an active adversary? ["fake ATM machine" / "credit card skimmer"]

Active adversary is able to first impersonate the server (i.e., interact with the client in an arbitrary manner) and afterwards, it tries to authenticate to the server (without further assistance from the client)

It is not known whether Schnorr's identification protocol is secure against active adversaries!