Password-based protocol <u>not</u> secure against eavesdropping adversary

(adversary sees vk and transcript of multiple interactions between honest prover + honest verifier)

One-time passwords (OTP) (SecurID tokens, Google authenticator, Duo)

<u>Construction 1</u>: Consider setting where verification key vk is <u>secret</u> (e.g., server has a secret)

- Client and server have a shared PRF key k and a counter (initialized to 0):

$$\begin{tikzcd} \text{client } (k, c) \arrow[rr, "{c', y' \leftarrow F(k,c)}"] \arrow[d] & & \text{server } (k, c) \\ c \leftarrow c+1 & & \end{tikzcd}$$

concretely: can intepret output as 6-digit number

check that $y' = F(k, c')$ and $c' > c$ (no replaying)

if successful, update $c \leftarrow c'$ } car key authentication

- <u>RSA SecurID</u>: stateful token (counter incremented by pressing button on token)
  - ↳ State is cumbersome — need to maintain consistency between client/server
- <u>Google Authenticator</u>: time-based OTP: counter replaced by current time window (e.g., 30-second window)

If PRF is secure ⟹ above protocol secure against eavesdroppers (but requires <u>server</u> secrets)

↳ can be problematic: RSA breached in 2011 and SecurID tokens compromised and used to compromise defense contractor Lockheed Martin

<u>Construction 2</u>: No server-side secrets (S/Key) "under composition"

- Relies on a hash function (should be one-way)
- Secret key is random input x and counter n;

  Verification key is $H^{(n)}(x) = \underbrace{H(H(\cdots H(x)\cdots))}_{n \text{ evaluations of } H}$

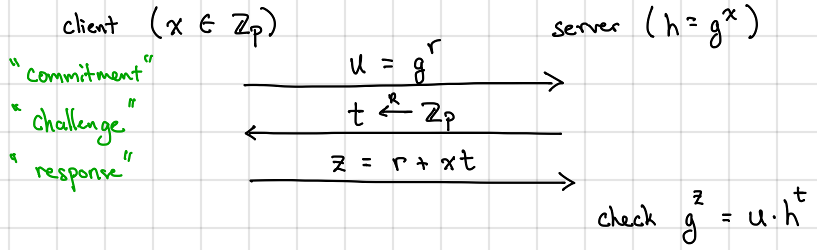
$$\begin{tikzcd} \mathsf{pwd}_n \arrow[d] & \mathsf{pwd}_{n-1} \arrow[d] & & & \mathsf{pwd}_2 \arrow[d] & \mathsf{pwd}_1 \arrow[d] & \\ x \arrow[r] & H(x) \arrow[r] & H^{(2)}(x) \arrow[r] & \cdots & H^{(n-2)}(x) \arrow[r] & H^{(n-1)}(x) \arrow[r] & H^{(n)}(x) = vk \end{tikzcd}$$

to verify y: check $H(y) \overset{?}{=} vk$

if successful, update $vk \leftarrow y$ } attacker has to invert H in order to authenticate

- Verification key can be public (credential is preimage of vk)
  - ↳ Can support <u>bounded</u> number of authentications (at most n) — need to update key after n logins
  - ↳ Output needs to be large (~80 bits or 128 bits) since password is the <u>input</u>/<u>output</u> to the hash function
- Naively, client has to evaluate H many times per authentication (~O(n) times)
  - ↳ Can reduce to O(log n) hash evaluations in an amortized sense by storing O(log n) entries along the hash chain

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


client $(x \in \mathbb{Z}_p)$
server $(h = g^x)$
"commitment"
"challenge"
"response"
$u = g^r$
$t \xleftarrow{R} \mathbb{Z}_p$
$z = r + xt$
check $g^z = u \cdot h^t$

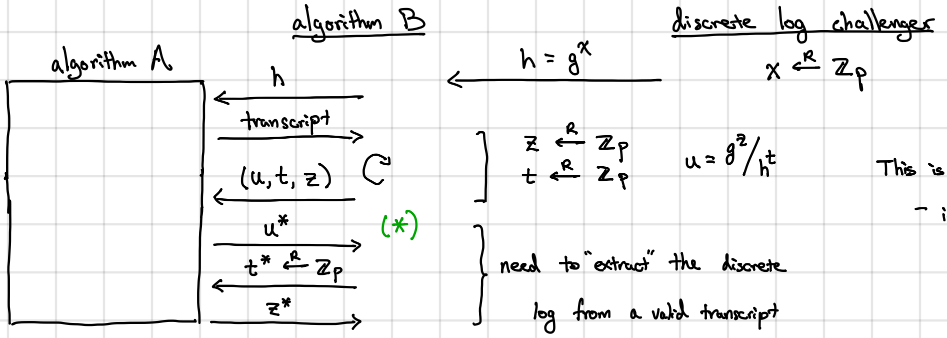
client will prove knowledge of x to the server

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



$z \xleftarrow{R} \mathbb{Z}_p$, $t \xleftarrow{R} \mathbb{Z}_p$ $\quad u = g^z / h^t$

need to "extract" the discrete log from a valid transcript

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 determin

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!